LLT.DLL

Schnittstellendokumentation

Micro-Optronic Messtechnik GmbH
Lessingstr. 14
01465 Langebrück OT. Dresden

Schnittstellendokumentation zur LLT.dll

Die LLT.dll ist eine DLL zum einfachen Integrieren des scanCONTROL's in eigene Anwendungen. Sie bildet eine Abstraktionsebene über dem direkten Ansprechen des Scanners per Firewire Bus oder der serielle Schnittstelle. Beim Design dieser DLL wurde besonderer Wert auf die Einfachheit der Schnittstelle und eine hohe Performance gelegt.
Um diese DLL mit möglichst vielen verschiedene Entwicklungsumgebungen und Compilern nutzen zu können, wurde die DLL Schnittstelle mit reinen C-Funktionen der „cdecl" und der „stdcall" Aufrufkonvention realisiert. Dadurch kann die DLL auch unter C, Delphi oder anderen Programmiersprachen genutzt werden (Bedingung dafür ist die Kompatibilität der verwendeten Datentypen). Für C++ Anwendungen gibt es eine zusätzliche Klasse mit deren Hilfe die C-Funktionen in Methoden einer Interface-Klasse gemappt werden.

In dieser Dokumentation wird nur die Einbindung der DLL in C++ beschrieben, die Einbindung in C oder in andere Programmiersprachen kann aus dieser Dokumentation abgeleitet werden.

1. Laden der DLL

Für das Laden einer DLL gibt es zwei verschiedene Möglichkeiten. Sie kann direkt beim Starten der Applikation geladen werden (statisch) oder später bei bedarf dynamisch. Das dynamische Laden ist meist günstiger, vor allem weil es eine bessere Fehlerbehandlung ermöglicht.

1.1. Statisches Laden

Beim statischen Laden der DLL in ein C oder C++ Projekt, wird die dazugehörige *.lib-Datei mit den Definitionen der DLL-Funktionen in das Projekt compiliert. Dabei ist es wichtig, dass immer die zu der verwendeten DLL-Version passende .lib-Datei verwendet wird. In den Header-Dateien `C_InterfaceLLT_2.h` und `S_InterfaceLLT_2.h` sind die zugehörigen Funktions-Deklarationen zu finden. Dabei steht der Präfix `s_` für „stdcall" und `c_` für „cdecl".

Für andere Programmiersprachen können anhand der `C_InterfaceLLT_2.h` oder der `S_InterfaceLLT_2.h` Importfunktionen für die DLL entwickelt werden.

Bei einer neuen DLL-Version muss das Projekt neu übersetzt werden, da sich die Einspringpunkte in der DLL ändern können.

1.2. Dynamisches Laden

Zum Laden der LLT.dll und Importieren ihrer Funktionen in C++-Projekte werden die zwei Klassen `CInterfaceLLT` und `CDllLoader` bereitgestellt.
Der `DllLoader` ist für das DLL handling (laden und Abfragen der Funktionspointer) zuständig. In der Interface-Klasse (`CInterfaceLLT`) sind alle Funktionen die die LLT.dll exportiert als Funktionspointer definiert. Sie können deshalb einfach als Methoden der Interface-Klasse aufgerufen werden.

```
#include "InterfaceLLT_2.h"
CInterfaceLLT* pInterfaceLLT;

...

//Anlegen der Interface-Klasse
pInterfaceLLT = new CInterfaceLLT();

//Testen ob es die gewuenschte Funktion gibt
if(pInterfaceLLT->m_pFunctions->CreateLLTFirewire != NULL)
{
  //Aufrufen von Funktionen in der LLT.dll
  pInterfaceLLT->CreateLLTFirewire();
}
```

Der herausragendste Vorteil dieser Interface-Klasse ist das dynamische Abfragen der Funktionen der DLL. Das heißt, es kann ohne das Projekt neu zu übersetzen eine neue DLL-Version eingesetzt werden. Zusätzlich kann noch abgefragt werden, ob die gewünschte Funktion in der DLL vorhanden ist. Dies ist aber nur für Funktionen, die nach dem ersten Release hinzugekommen sind, notwendig.
Sollen neue Funktionen der LLT.dll verwendet werden, muss das Projekt mit der aktuellen Interface-Klasse neu übersetzt werden.

Zusätzlich kann dem Konstruktor der Interface-Klasse noch der Name der zu ladenden DLL (mit Pfad) und ein Pointer auf eine `bool`-Variable übergeben werden, welche einen Fehler beim Laden der DLL signalisiert.

2. Beschreibung der einzelnen Funktionen

Die Funktionen der LLT.dll gliedern sich in mehrere Funktionsgruppen:

Funktionsgruppe	Beschreibung
Instanz-Funktionen	Zum Erstellen einer scanCONTROL-Instanz mit Firewire- oder serieller Schnittstellenunterstützung und zum Löschen dieser Instanz
Auswahl-Funktionen	Zum Auswählen eines scanCONTROL's
Verbindungs-Funktionen	Verbinden und Schließen einer Verbindung mit einem scanCONTROL
Identifikations-Funktionen	Abfragen des Namens und der Version
Eigenschafts-Funktionen	Abfragen und Setzen von Eigenschaften (z.B. Belichtungszeit, Seriennummer ...)
Spezielle Eigenschafts-Funktionen	Abfragen und Setzen von speziellen Eigenschaften
Register Funktionen	Zum Registrieren von einem Callback und einer Error-Message
Profilübertragungs-Funktionen	Übertragen von Profilen
Is Funktionen	Zum Abfragen von verschiedenen Zuständen und Verbindungen
PartialProfile Funktionen	Einschränkung des zu übertragenden Profils auf dem scanCONTROL
Time Funktionen	Auswerten des Timestamps
Postprocessing Funktionen	Lesen und Schreiben der Postprocessing-Parameter
File Funktionen	Laden und Speichern von Profilen
Spezielle CMM-Trigger Funktionen	Starten und Stoppen der Profilübertragung inklusive des CMM-Triggers
Fehlerwert Konvertierungs-Funktion	Konvertieren des Fehlerwertes von Funktionen in Text

Für alle die hier beschriebenen Funktionen sind immer die Parameter für die `CInterfaceLLT`-Klasse angegeben. Wird diese Klasse nicht verwendet hat jede dieser Funktionen einen zusätzlichen ersten Parameter (`Instanzhandle`). Alle andere Parameter der Funktionen verschieben sich um eins nach hinten (siehe die Datei `C_InterfaceLLT_2.h` oder `S_InterfaceLLT_2.h`).
Dieser zusätzliche erste Parameter dient zur Unterscheidung der Verschiedenen scanCONTROL-Instanzen in der DLL.
In der `CInterfaceLLT`-Klasse wird dieser Parameter automatisch eingefügt.

2.1. Funktionen der LLT.dll bis Version 2.0.0

Alle Funktionen der LLT.dll bis Version 2.0.0 bleiben erhalten. Die neue DLL-Version kann ohne Änderungen an bestehender Software eingesetzt werden. Neue Funktionen werden aber nur noch zu dem neuen Interface hinzugefügt.

In der folgenden Tabelle sind alle nicht mehr vorhandenen Funktionen des alten Interfaces mit ihren äquivalenten neuen Funktionen des neuen Interfaces aufgelistet. Alle Funktionen den neuen Interfaces besitzen jetzt ein `int` als Rückgabewert. Die einzelnen Rückgabewerte werden in den nächsten Kapiteln beschrieben.

Alter Funktionsname	Neuer Funktionsname	Kapitel
GetDeviceVendorName	GetDeviceName	2.7
Get / SetSerialNr	Get / SetFeature	2.8.1
Get / SetLaserPower	Get / SetFeature	2.8.2
Get / SetMeasuringField	Get / SetFeature	2.8.3
Get / SetTrigger	Get / SetFeature	2.8.4
Get / SetTrigger_2	Get / SetFeature	2.8.4
Get / SetShutter	Get / SetFeature	2.8.5
Get / SetDeadTime	Get / SetFeature	2.8.6
Get / SetExposure	Get / SetFeature	2.8.7
Get / SetThreshold	Get / SetFeature	2.8.8
Get / SetVideoMonitor	Get / SetFeature	2.8.9
Get / SetAnalogFrequency	Get / SetFeature	2.8.10
Get / SetAnalogOutputsModes	Get / SetFeature	2.8.11
Get / SetCMMTrigger	Get / SetFeature	2.8.12
Get / SetLargePictures	entfällt	
Get / SetMainPeak	Get / SetMainReflection	2.9.2
RegisterCallback_2	RegisterCallback	2.10.1
GetProfile (für Firewire)	TransferProfiles	2.11.1
EnableShots	TransferProfiles	2.11.1
OneShot	MultiShot	2.11.2
Timestamp2Double	Timestamp2TimeAndCount	2.14
LoadProfiles_2	LoadProfiles	2.16.2
StartTransmissionAndCmmTrigger_2	StartTransmissionAndCmmTrigger	2.17

2.2. Allgemeine Rückgabewerte der Funktionen

Alle Funktionen des Interfaces geben einen `int` Wert als Rückgebewert zurück. Ist der Rückgabewert einer Funktion größer oder gleich `GENERAL_FUNCTION_OK` bzw. `'1'` so war die Funktion erfolgreich, ist der Rückgabewert `GENERAL_FUNCTION_NOT_AVAILABLE` bzw. `'0'` oder negativ so ist ein Fehler aufgetreten.
Eine Besonderheit gibt es bei einigen Funktionen, die auch `GENERAL_FUNCTION_CONTAINER_MODE_HEIGHT_CHANGED` bzw. `'2'` zurückgeben können. Tritt dieser Rückgabewert auf, hat sich die Größe des Bildes im Container-Mode geändert (siehe Kapitel 2.11 „Profilübertragungs-Funktionen“).

Zur Unterscheidung der einzelnen Rückgabewerte stehen mehrere Konstanten zu Verfügung. In der folgenden Tabelle sind alle allgemeinen Rückgabewerte aufgeführt, die von Funktionen zurückgegeben werden können. Für die einzelnen Funktionsgruppen kann es zusätzlich noch spezielle Rückgabewerte/Fehlerwerte geben.

Konstante für den Rückgabewert	Wert	Beschreibung
GENERAL_FUNCTION_CONTAINER_MODE_HEIGHT_CHANGED	2	Funktion erfolgreich ausgeführt, aber die Bild-Größe für den Container-Mode wurde verändert
GENERAL_FUNCTION_OK	1	Funktion erfolgreich ausgeführt
GENERAL_FUNCTION_NOT_AVAILABLE	0	Diese Funktion ist nicht verfügbar, ev. eine neue DLL verwenden oder in den Firewire-Mode wechseln
ERROR_GENERAL_WHILE_LOAD_PROFILE	-1000	Funktion konnte nicht ausgeführt werden, da das Laden von Profilen aktiv ist
ERROR_GENERAL_NOT_CONNECTED	-1001	Es besteht keine Verbindung zum scanCONTROL -> Connect aufrufen
ERROR_GENERAL_DEVICE_BUSY	-1002	Die Verbindung zum scanCONTROL ist gestört oder getrennt -> neu Verbinden und Anschluss des scanCONTROL's überprüfen
ERROR_GENERAL_WHILE_LOAD_PROFILE_OR_GET_PROFILES	-1003	Funktion konnte nicht ausgeführt werden, da entweder das Laden von Profilen oder die Profilübertragung aktiv ist
ERROR_GENERAL_WHILE_GET_PROFILES	-1004	Funktion konnte nicht ausgeführt werden, da die Profilübertragung aktiv ist
ERROR_GENERAL_GET_SET_ADDRESS	-1005	Die Adresse konnte nicht gelesen oder geschrieben werden. Eventuell wird eine zu alte Firmware verwendet.
ERROR_GENERAL_POINTER_MISSING	-1006	Ein benötigter Pointer ist NULL
ERROR_GENERAL_WHILE_SAVE_PROFILES	-1007	Funktion konnte nicht ausgeführt werden, da das Speichern von Profilen aktiv ist

2.3. Instanz-Funktionen

Die LLT.dll unterstützt die Kommunikation zum Scanner über die serielle Schnittstelle und über Firewire. Nach dem Laden der DLL muss entweder mit CreateLLTFirewire() oder mit CreateLLTSerial() ein Device in der DLL angelegt werden. Bei Erfolg geben diese Funktionen GENERAL_FUNCTION_OK zurück.

Wird die CInterfaceLLT-Klasse nicht verwendet, geben diese beiden Funktionen statt GENERAL_FUNCTION_OK oder GENERAL_FUNCTION_NOT_AVAILABLE ein Instanzhandle für die in der LLT.dll angelegte interne Instanz zurück. Dieses Handle muss allen weiteren Funktionen als erster Parameter mit übergeben werden.
Ist dieses Instanzhandle 0 oder 0xffffffff ist das erstellen eines Devices fehlgeschlagen.

Mit Hilfe der Funktion DelDevice() muss vor dem Entladen der DLL das angelegte Device wieder gelöscht werden (dies wird in der CInterfaceLLT Klasse automatisch gemacht).

Bei einem DelDevice() bleiben alle Parameter die im scanCONTROL eingestellt wurden erhalten, außer die „Resolution“ (siehe Kapitel 2.9.6), die „Profile config“ (siehe Kapitel 2.9.5), die „Packet size“ (siehe Kapitel 2.9.4) und der „Buffer count“ im Treiber (siehe Kapitel 2.9.1).

2.4. Parallelbetrieb mehrerer scanCONTROL's

Um mehrere scanCONTROL's parallel in einem Programm zu nutzen gibt es zwei Möglichkeiten, abhängig von der gewählten Methode zum Laden der LLT.dll.

Statisches Laden:

Pro verbundenen scanCONTROL muss jeweils die Funktion `CreateLLTFirewire()` oder `CreateLLTSerial()` aufgerufen werden. Durch die unterschiedlichen zurückgegebenen `Instanzhandle` können die verschiedenen scanCONTROL's bzw. Instanzen unterschieden werden.

Dynamisches Laden:

Pro verbundenen scanCONTROL muss jeweils eine Instanz der `CInterfaceLLT` – Klasse angelegt werden. Die Instanzen der Klasse verwalten selbstständig den `Instanzhandle` für jedes scanCONTROL.

Sollen Callbacks verwendet werden können die unterschiedlichen Instanzen sich einen Callback teilen (siehe Kapitel 2.10.1 „Callback").

```
#include <vector>

void main()
{
  pInterfaceLLT_1 = new CInterfaceLLT();
  pInterfaceLLT_2 = new CInterfaceLLT();

  //Erstellen von zwei Firewire Devices/Instanzen
  pInterfaceLLT_1->CreateLLTFirewire();
  pInterfaceLLT_2->CreateLLTFirewire();

  std::vector<DWORD> FirewireInterfaces(6);

  //Auslesen der verfuegbaren Interfaces
  int FirewireInterfaceCount = pInterfaceLLT_1->GetDeviceInterfaces(
    &FirewireInterfaces[0], FirewireInterfaces.size());

  if(FirewireInterfaceCount < 2)
  {
    //Es ist nur 1 scanCONTROL verbunden
    return;
  }

  //Setzen der verschiedenen Interfaces
  pInterfaceLLT_1->SetDeviceInterface(FirewireInterfaces[0], 0);
  pInterfaceLLT_2->SetDeviceInterface(FirewireInterfaces[1], 0);

  //Verbinden der zwei scanCONTROL's
  pInterfaceLLT_1->Connect();
  pInterfaceLLT_2->Connect();

  //Registrieren des Callbacks
  pInterfaceLLT_1->RegisterCallback(STD_CALL, (void*)NewProfile, 0));
  pInterfaceLLT_2->RegisterCallback(STD_CALL, (void*)NewProfile, 1));
  . . .
}

//Callback
void _stdcall NewProfile (const unsigned char* Data, unsigned int Size,
                          void* UserData)
{
  switch((int)UserData)
  {
    case 0:
    {
      //Callback Aufgerufen von LLT_1
    }
    case 1:
    {
      //Callback Aufgerufen von LLT_2
    }
  }
}
```

2.5. Auswahl-Funktionen

Mit Hilfe der Auswahlfunktionen kann eine Auswahl des Device-Interfaces erfolgen. Zum Beispiel kann damit die verwendete serielle Schnittstelle oder die Firewire-Node-ID ausgewählt werden.

```
int GetDeviceInterfaces(unsigned int *pInterfaces, unsigned int nSize)
```

Abfrage der Device-Interfaces an die ein scanCONTROL angeschlossen ist. Der `Interfaces`-Parameter ist ein Feld aus Integer-Variablen, in das die Interface-Nummern eingetragen werden. `Size` gibt dabei die Größe des Feldes an.
Bei der Kommunikation über Firewire werden die Node-IDs, die einem scanCONTROL zugeordnet sind in das Feld eingetragen, bei einer Verbindung über die serielle Schnittstelle die Nummern aller benutzbaren Com-Ports (Port 1 bis 16).
Der Rückgabewert gibt die Anzahl der gefundenen und in das `Interfaces`-Feld eingetragenen Device-Interfaces zurück.
Ist der Rückgabewert kleiner oder gleich `GENERAL_FUNCTION_NOT_AVAILABLE` kann er einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
`ERROR_GETDEVINTERFACES_WIN_NOT_SUPPORTED`	-250	Funktion steht nur unter Windows 2000 oder höher zur Verfügung
`ERROR_GETDEVINTERFACES_REQUEST_COUNT`	-251	Die Größe des übergebenen Feldes ist zu klein

```
void SetDeviceInterface(unsigned int nInterface, int nAdditional)
```

Setzen der Nummer eines Interfaces und Übergabe eines zusätzlichen Parameters. Dieser wird zurzeit nur bei einer Verbindung über die serielle Schnittstelle verwendet und muss 115 000 betragen (als Baudrate der seriellen Schnittstelle).
Ist der Rückgabewert kleiner oder gleich `GENERAL_FUNCTION_NOT_AVAILABLE` kann er einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
`ERROR_GETDEVINTERFACES_CONNECTED`	-252	Das scanCONTROL ist verbunden, `Disconnect();` aufrufen

```
unsigned int nAvailibalInterfaces[20];

int nInterfaces = pInterfaceLLT->GetDeviceInterfaces
                                        (nAvailableInterfaces, 20);
if(nInterfaces > GENERAL_FUNCTION_NOT_AVAILABLE)
{
  pInterfaceLLT->SetDeviceInterface(nAvailableInterfaces[0], 115000);
}
```

2.6. Verbindungs-Funktionen

Mit Hilfe der Verbindungsfunktionen kann eine Verbindung zu einem ausgewählten scanCONTROL aufgenommen oder beendet werden.

```
int Connect()
```

Verbinden der LLT.dll mit einem ausgewählten scanCONTROL. Wurde vorher kein scanCONTROL mit Hilfe von `SetDeviceInterface` ausgewählt, wird bei Firewire der Scanner mit der Node-ID 0 oder bei einer seriellen Verbindung der Com-Port 1 ausgewählt.

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
`ERROR_CONNECT_LLT_COUNT`	-300	Es ist kein scanCONTROL am Computer angeschlossen oder der Treiber ist nicht korrekt installiert
`ERROR_CONNECT_SELECTED_LLT`	-301	Das gewählte Interface ist nicht verfügbar -> ein neues Interface mit `SetDeviceInterface` wählen
`ERROR_CONNECT_ALREADY_CONNECTED`	-302	Mit dieser ID ist schon ein scanCONTROL verbunden
`ERROR_CONNECT_LLT_NUMBER_ALREADY_USED`	-303	Das gewünschte scanCONTROL wird schon von einer anderen Instanz verwendet -> mit `SetDeviceInterface` ein anderes scanCONTROL auswählen
`ERROR_CONNECT_SERIAL_CONNECTION`	-304	Es konnte sich nicht per serieller Schnittstelle mit dem scanCONTROL verbunden werden -> mit `SetDeviceInterface` ein anderes scanCONTROL auswählen

```
int Disconnect()
```

Trennen einer Verbindung zu einem scanCONTROL.
Bei einem Disconnect bleiben alle Parameter die im scanCONTROL eingestellt wurden erhalten, außer die „Resolution“ (siehe Kapitel 2.9.6), die „Profile config“ (siehe Kapitel 2.9.5), die „Packet size“ (siehe Kapitel 2.9.4) und der „Buffer count“ im Treiber (siehe Kapitel 2.9.1).

2.7. Identifikations-Funktionen

```
int GetDeviceName(char *pDevName, unsigned int nDevNameSize,
                  char *pVenName, unsigned int nVenNameSize)
```

Abfragen des Gerätenamens und des Herstellernamens des scanCONTROL’s. Der Gerätenamen ist zum Beispiel „LLT2800-100(000)v17-C2“. In dem Gerätenamen sind der Messbereich (in diesem Fall 100mm), die Option (000) und die Softwareversion des Scanners codiert. Der Herstellername ist immer „micro-optronic“.
Wird einer der Namen nicht benötigt, kann anstatt eines Zeigers auf den Puffer auch eine `NULL` übergeben werden.

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_GETDEVICENAME_SIZE_TOO_LOW	-1	Die Größe eines der Puffers ist zu klein
ERROR_GETDEVICENAME_NO_BUFFER	-2	Es wurde kein Puffer übergeben

```
char DeviceName[100];

memset(DeviceName, 0, sizeof(DeviceName));
if(pInterfaceLLT->GetDeviceName(DeviceName, sizeof(DeviceName),
                                NULL, NULL) > GENERAL_FUNCTION_NOT_AVAILABLE)
{
  //Verarbeiten des Geraetenamens
}
```

```
int GetLLTVersions(unsigned int *pDSP, unsigned int *pFPGA1,
                   unsigned int *pFPGA2)
```

Abfragen der Software-Versionen des Scanners. Sie werden automatisch aus dem Gerätenamen extrahiert.

2.8. Eigenschafts-Funktionen

Mit den Eigenschafts-Funktionen können die verschiedenen Eigenschaften des Scanners gelesen oder geschrieben werden. Es gibt eine `GetFeature`- und eine `SetFeature`-Funktion. Über einen Parameter kann die zu lesende oder schreibende Eigenschaft ausgewählt werden.

```
int GetFeature(DWORD Function, DWORD *pValue);
int SetFeature(DWORD Function, DWORD Value);
```

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_SETGETFUNCTIONS_WRONG_FEATURE_ADRESS	-155	Die Adresse der gewählten Eigenschaft ist falsch

In folgender Tabelle sind alle verfügbaren Eigenschaften aufgeführt:

Konstante für die Eigenschaft	Adresse	Beschreibung
FEATURE_FUNCTION_SERIAL	0xf0000410	Serial
FEATURE_FUNCTION_LASERPOWER	0xf0f00824	Laser power
FEATURE_FUNCTION_MEASURINGFIELD	0xf0f00880	Measuring field
FEATURE_FUNCTION_TRIGGER	0xf0f00830	Trigger
FEATURE_FUNCTION_SHUTTERTIME	0xf0f0081c	Shutter time
FEATURE_FUNCTION_IDLETIME	0xf0f00800	Idle time
FEATURE_FUNCTION_PROCESSINGPROFILEDATA	0xf0f00804	Processing profile data
FEATURE_FUNCTION_THRESHOLD	0xf0f00810	Threshold
FEATURE_FUNCTION_MAINTENANCEFUNCTIONS	0xf0f0088c	Maintenance functions
FEATURE_FUNCTION_ANALOGFREQUENCY	0xf0f00828	Analog frequency
FEATURE_FUNCTION_ANALOGOUTPUTMODES	0xf0f00820	Analog output modes
FEATURE_FUNCTION_CMMTRIGGER	0xf0f00888	CMM trigger
FEATURE_FUNCTION_REARRANGEMENT_PROFILE	0xf0f0080c	Rearrangement profile
FEATURE_FUNCTION_PROFILE_FILTER	0xf0f00818	Profile filter

2.8.1. Serial

Lesen der Seriennummer. Diese Eigenschaft kann nur gelesen werden.

2.8.2. Laser power

Abfragen oder Setzen der Laserleistung. Die Laserleistung ist in den niedrigsten 3 Bits folgendermaßen codiert:

Bit	Funktion								
2..0	Wert	0	1	2	3	4	5	6	7
	LASER_OFF nicht verbunden	aus	aus	aus	nicht erlaubt	aus	red	std	nicht erlaubt
	LASER_OFF verbunden	aus	red	std		aus	aus	aus	

Die Laser-Zustände sind in der folgenden Tabelle erläutert:

Status	Beschreibung
aus	Laser ist aus
red	Laser leuchtet mit reduzierter Leistung
std	Laser leuchtet mit Standard (voller) Leistung

Das 2. Bit der Laserpower ist der LaserOff-Eingang. Damit kann ausgewählt werden, ob der Laser bei offenen LaserOff-Eingang an oder aus ist. Der LaserOff-Eingang kann über die Synchron-Buchse angeschlossen werden.

Das erste Profil nach dem Umschalten der Laserleistung kann korrupt sein. Es ist daher sinnvoll erst das zweite Profil danach zu verarbeiten.

2.8.3. Measuring field

Abfragen oder Setzen des Messfeldes. Die Messfeldnummer kann zwischen 0 und 95 liegen. Für weitere Informationen siehe Kapitel 8.2 „Messfeldauswahl und Kalibrierung“ und 11.3 „Unterstützte Messfelder“ in der Bedienungsanleitung des Scanners.
Die Auswahl des Messfeldes hat Einfluss auf die maximale Profilfrequenz des scanCONTROL’s (siehe Kapitel „Maximum Frequencies of Profile Measurements“ in der „QuickReference.html“).

Das erste Profil nach dem Umschalten des Messfeldes kann korrupt sein. Es ist daher sinnvoll erst das zweite Profil danach zu verarbeiten.

2.8.4. Trigger

Abfragen oder Setzen des Triggers.

Bit	**Beschreibung**
19..16	Unterstützte Trigger Modes: 0 Flanken-Mode, der Shutter wird mit der führenden Flanke geöffnet und nach der Shutter-Time automatisch wieder geschlossen 1 Pulse-Mode, der Shutter ist solange offen, wie der Trigger-Impuls andauert
24	Polarität: 0 Low aktiv 1 High aktiv
25	Trigger-Quelle: Interner Trigger Externer Trigger

Die Polarität beschreibt die Polarität der externen Trigger Quelle. Ist die Polarität `Low aktiv`, wird bei jedem `High` zu `Low` Übergang ein Profil erzeugt. Für `High aktiv` ist die Funktionalität analog.

Das erste Profil nach dem Umschalten des Trigger-Modes kann korrupt sein. Es ist daher sinnvoll erst das zweite Profil danach zu verarbeiten.

Wird der Scanner über den Firewire-Bus betrieben und die Triggerquelle auf „extern“ gesetzt, ist es sinnvoll die Pufferanzahl auf 2 zu stellen, da es sonst zu erheblichen Profilausfällen kommen kann. Dies ist nicht nötig, wenn sichergestellt ist, dass innerhalb von 10 Sekunden mindestens so viele Triggerimpulse ausgelöst werden wie Puffer verwendet werden.

2.8.5. Shutter time

Abfragen oder Setzen der Belichtungszeit. Sie kann zwischen 1 und 4095 liegen. Ein Zählwert entspricht dabei 10 µs. Aus der Summe von Belichtungszeit und Totzeit (siehe Kapitel 2.8.6 „Idle time“) kann die Profilfrequenz berechnet werden.

Bit	Funktion
11..0	Belichtungszeit in 10µs Schritten (1 bis 4095)
24	Automatische Belichtungszeitregelung (0 = deaktiviert, 1 = aktiviert)

Der Bildsensor unterstützt überlappendes auslesen. Wenn die Belichtungszeit zum Auslesen des Sensors reicht, kann die Todzeit auf 20 µs verkleinert werden. Die Profilrate hängt von der Auflösung (Anzahl der Punkte pro Profil) und dem Messfeld ab. Mehr Details sind im Kapitel 8.2 „Messfeldauswahl und Kalibrierung“ und Kapitel 11.3 „Unterstützte Messfelder“ in der Bedienungsanleitung des Scanners zu finden. Eine Tabelle mit den zu erreichenden Profilraten bei den unterschiedlichen Auflösungen und Paketgrößen ist in der „QuickReference.html“ zu finden.

Profilrate = 1000 / (Belichtungszeit in ms + Totzeit in ms)

Mit Hilfe des 24. Bits kann zusätzlich eine automatische Belichtungszeitregelung eingeschaltet werden. Sie regelt die Belichtungszeit in Abhängigkeit vom Messobjekt. Die unteren 12 Bit geben dabei einen Vorgabewert vor, welcher verwendet wird wenn kein Messobjekt gesehen wird. Bei der automatischen Belichtungszeitregelung ändert sich nicht die Profilrate.

2.8.6. Idle time

Abfragen oder Setzen der Totzeit. Sie kann zwischen 1 und 4095 liegen. Ein Zählwert entspricht dabei 10 µs. Aus der Summe von Totzeit und Belichtungszeit (siehe Kapitel 2.8.5 „Shutter time“) kann die Profilfrequenz berechnet werden.

Bit	Funktion
11..0	Totzeit in 10µs Schritten (1 bis 4095)

Profilrate = 1000 / (Belichtungszeit in ms + Totzeit in ms)

Ist die automatische Belichtungszeitregelung aktiv (siehe Kapitel 2.8.5 „Shutter time“) wird die Totzeit automatisch angepasst, damit die Profilfrequenz stabil bleibt.

2.8.7. Processing profile data

Bit	Funktion
0	Auflösung der Entfernung: 0 niedrig (schnell) 1 hoch (langsam)
1	Kalibrierung (konvertiert Spalten und Zeilen in Millimeter): 0 deaktiviert (schneller, ist aber nur für spezielle Aufgaben sinnvoll) 1 aktiviert
2..4	Berechnungen bei mehreren Reflektionen: 0 alle Reflektionen berechnen 1 nur die jeweils erste Reflektion berechnen 2 nur die jeweils letzte Reflektion berechnen und an die erste Position kopieren 3 nur die Reflektion mit der jeweils höchsten integralen Intensität berechnen und an die erste Position kopieren 4 nur die Reflektion mit der jeweils höchsten maximalen Intensität berechnen und an die erste Position kopieren
5	Aktivierung des Postprocessing (0 = deaktiviert, 1 = aktiviert)
6	Umkehren der Abstandskoordinate (Z-Werte) (0 = deaktiviert, 1 = aktiviert)
7	Umkehren der Positionskoordinate (X-Werte) (0 = deaktiviert, 1 = aktiviert)
8	Verzögern der automatischen Belichtungszeitregelung: 0 Die neue Belichtungszeit wird nach der „Berechnungen bei mehreren Reflektionen“ berechnet 1 Die neue Belichtungszeit wird nach allen Berechnungen eines Profils inklusive dem Postprocessings berechnet
10..9	Ausrichtung des nächsten Belichtungsintervalls (ist nur sinnvoll in Kombination mit der automatischen Belichtungszeitregelung): 0 Gleiches Ausrichten der Belichtungszeit-Mitten 1 Gleiches Ausrichten der Belichtungszeit-Enden (rechte Ecke) 2 Gleiches Ausrichten der Belichtungszeit-Anfängen (linke Ecke) 3 Kein Ausrichten der Belichtungszeiten, schnellste Reaktion

2.8.8. Threshold

Schwelle zur Auswahl von Reflektionen. Bei Targets mit mehreren Reflektionen kann das erhöhen der Schwelle zu besseren Ergebnissen führen.

Bit	Funktion
0..9	Schwellwert zur Erkennung von Reflektionen (0 bis 1023)
10	Unterdrückung von Fremdlicht (0 = deaktiviert, 1 = aktiviert)
11	Video filter (0 = deaktiviert, 1 = aktiviert)

2.8.9. Maintenance functions

Bit	Funktion
0	Video Ausgang (0 = deaktiviert, 1 = aktiviert)
1	Parameter loopback (0 = deaktiviert, 1 = aktiviert)
2	Head and tail (0 = deaktiviert, 1 = aktiviert)
3	Interner Zähler für positive Flanken (0 = deaktiviert, 1 = aktiviert)
4	Interner Zähler für negative Flanken (0 = deaktiviert, 1 = aktiviert)

Video Ausgang:

Aktiviert die Videoausgabe.

Parameter loopback:

Aktiviert das Parameter loopback. Mit dieser Funktion werden alle Parameter des scanCONTROL's in das 0. Moment und 1. Moment des 4. Streifens kopiert.

Head and tail:

Errechnet für jeden Punkt Koordinate jeweils Anfang, Mitte und Ende einer Reflexion. Die Mitten werden in die erste Reflexion, die Anfangspunkte in die zweite Reflexion und die Endpunkte in die dritte Reflexion eingetragen.

Ist nur mit der reduzierten „Auflösung der Entfernung" und „alle Reflektionen berechnen" sinnvoll (siehe Kapitel 2.8.7 „Processing profile data"). Zum Aufbau eines Profils siehe Kapitel 3.1 „Beschreibung der Profil-Daten".

Interner Zähler für Flanken:

Löscht den internen Zähler und startet mit dem Zählen der positiven, negativen oder beiden Flanken. Der aktuelle Zählerstand wird in den Timestamp jedes Profils nach der Belichtung kopiert (für eine Beschreibung des Timestamps siehe Kapitel 3.4 „Beschreibung des Timestamps").

Die Ausgabe des Zählerstandes im Timestamp ist immer die Addition aus dem internen Zählerstand zu Begin und zum Ende der Integration -> er zeigt immer die doppelte Anzahl der Flanken an. Wird der ausgegebene Zählerstand durch zwei dividiert erhält man den interpolierten Zählerstand zur Mitte der Integration.

Der Zähler ist optional.

```
#include <vector>
//Aktivieren des internen Zaehlers, Aufloesung setzen
pInterfaceLLT->SetFeature(FEATURE_FUNCTION_MAINTENANCEFUNCTIONS,
                          0x00000010);
pInterfaceLLT->SetResolution(256);

//Aktivieren der Profiluebertragung und einen Moment warten (auf Profile)
pInterfaceLLT->TransferProfiles(NORMAL_TRANSFER, true);
Sleep(500);

//Erstellen eines Puffers fuer ein Profil in PURE_PROFILE Mode und abholen
//eines Profils
unsigned int ProfSize = 256*4+16;
std::vector<unsigned char> vProfile(ProfSize);
if(pInterfaceLLT->GetActualProfile(&vProfile[0], ProfSize, PURE_PROFILE,
                                   NULL) != ProfSize)
{
  //Das scanCONTROL sendet keine Profile
  return;
}
unsigned int LastCount = 0, OverflowCount = 0, TempCount;

pInterfaceLLT->Timestamp2CmmTriggerAndInCounter(&vProfile[ProfSize-16],
                                        TempCount, NULL, NULL, NULL);

//Testen ob der 16 bit Zähler uebergelaufen ist
if(TempCount < (LastCount & 0x0000ffff))
{
  OverflowCount += 1;
}
//Erstellen des echten Zählerstandes mit allen Ueberlaeufen
LastCount = TempCount + 0x00010000 * OverflowCount;
```

2.8.10. Analog frequency

Abfragen oder Setzen der Frequenz für den Analogausgang. Die Frequenz kann zwischen 0 und 150 eingestellt werden, wobei der Zählwert der Frequenz in kHz entspricht. Bei einer Einstellung von 0 kHz wird der Analogausgang abgeschaltet, was bei Profilfrequenzen größer 500 Hz empfehlenswert ist, um einen Überlauf bei der Analogausgabe zu vermeiden.

Bit	Funktion
7..0	Frequenz der Analogausgänge in kHz (0 bis 150)

2.8.11. Analog output modes

Einstellen der Analog output modes. Es können z.B. die Spannungsbereiche und die Polarität der analogen Ausgänge umgeschaltet werden.

Bit	Funktion
0..1	Spannungsbereich der Abstands-Koordinate: 0 ±10V 1 ±5V 2 -10V..0V 3 0..10V
2	Tauscht die Polarität der Abstands-Koordinate (0 = deaktiviert, 1 = aktiviert)
3	Hält den letzten gültigen Wert (beide Koordinaten) (0 = deaktiviert, 1 = aktiviert)
4..5	Spannungsbereich der Positions-Koordinate 0 ±10V 1 ±5V 2 -10V..0V 3 0..10V
6	Tauscht die Polarität der Positions-Koordinate (0 = deaktiviert, 1 = aktiviert)

2.8.12. CMMTrigger

Konfiguration des optionalen CMM-Triggers. Die Konfiguration des CMM-Triggers besteht aus 4 Befehlswörtern. Diese Befehlswörter müssen nacheinander geschrieben werden. Es kann auch jeweils nur eines der Befehlswörter ausgetauscht werden, ohne die anderen schreiben zu müssen.
Zu beachten ist, dass die Befehlswörter 2 bis 4 immer nur bei ausgeschaltetem (Divisor = 0) CMM-Trigger geschrieben werden dürfen.
Ist der Divisor (1. Befehlswort) 0, ist der CMM-Trigger deaktiviert.

Bits 11..10	Bit 9	Bit 8	Bits 7..0
0	Polarität 0 Low aktiv 1 High aktiv	nicht benutzt	Divisor (unsigned integer), 0 deaktiviert den Trigger
1	nicht benutzt	ungefähres Impuls/Pausenverhältnis (signed integer), -1 ¦ 0 ¦ 1 meint 1:1, -2 meint 1:2, 2 meint 2:1	
2	Phasenkorrektur (signed integer), in 0.5µs Schritten, muss kleiner als die Shutter/Idle time sein		
3	nicht benutzt	nicht benutzt	Ausgangsnummer, nur 0 ¦ 1 ¦ 2

Der CMM-Trigger muss nach dem deaktivieren nicht neu konfiguriert werden, es reich wenn nur das 1. Befehlswort neu geschrieben wird.

Beim auslesen des CMM-Triggers kann immer nur das zuletzt geschriebene Befehlswort zurückgelesen werden.

Ist der CMM-Trigger aktiv wird im Timestamp ein zusätzlicher Zähler eingeblendet. Eine Dokumentation des Timestamps finden Sie im Kapitel 3.4 „Beschreibung des Timestamps“.

2.8.13. Rearrangement profile

Im Container-Mode werden mehrere Profile zu einem Container/Bild zusammengefasst. In den Spalten stehen die einzelnen Punkte mit den ausgewählten Eigenschaften und in den Zeilen die einzelnen Profile.
Zum Beispiel:

	Z Punkt 1	**...**	**Z Punkt n**	**X Punkt 1**	**...**	**X Punkt n**
Profil 1						
Profil 2						
Profil 3						
Profil 4						
Profil 5						

Die Auflösung eines Punktes beträgt 16 Bit. Dieser Container kann als 16 Bit Graustufen-Bitmap direkt angezeigt werden, oder es können mit Bildverarbeitungsalgorithmen Merkmale extrahiert werden.
Dadurch dass mehrere Profile zu einem Bild zusammengefasst werden, wird auch bei hohen Profilraten der PC entlastet.
Der Aufbau der Container ist in Kapitel 3.2 „Beschreibung des Datenformates CONTAINER“ beschrieben.

Um korrekte Ergebnisse beim Aneinanderreihen von Profile zu bekommen wird empfohlen die Profile vorher mit dem „Profile filter“ auf äquidistante Positions-Koordinaten zu bringen (siehe Kapitel 2.8.14 „Profile filter“).

Beispiele zum Übertragen von Containern sind im Kapitel 4 „LLT2800Samples“ zu finden.

Diese Funktion ist erst ab DSP-Version 17 vorhanden.

Bit	Funktion
23	extrahiert die Felder des 4. Streifens (0 = deaktiviert, 1 = aktiviert)
22	extrahiert die Felder des 3. Streifens (0 = deaktiviert, 1 = aktiviert)
21	extrahiert die Felder des 2. Streifens (0 = deaktiviert, 1 = aktiviert)
20	extrahiert die Felder des 1. Streifens (0 = deaktiviert, 1 = aktiviert)
19	Verbinden von aufeinander folgenden Profilen (erlaubt größere Pakete), bei 512 Punkten/Profil und 1024 Punkten/Profil gibt es Einschränkungen (0 = deaktiviert, 1 = aktiviert)
18	Datenformat der Felder: (0 = unsigned, 1 = signed)
17..16	Reserviert
15..12	Punkte pro Profil: 6 64 Punkte 7 128 Punkte 8 256 Punkte 9 512 Punkte 10 1024 Punkte
11	Timestamp einfügen (0 = deaktiviert, 1 = aktiviert)
10	Einfügen eines leeren Feldes, damit der Timestamp keine wertvollen Informationen überschreibt (0 = deaktiviert, 1 = aktiviert)
9	Reserviert
8	extrahiert Moment 1 (höheren 16 Bits) (0 = deaktiviert, 1 = aktiviert)
7	extrahiert Moment 1 (niederen 16 Bits) (0 = deaktiviert, 1 = aktiviert)
6	extrahiert Moment 0 (höheren 16 Bits) (0 = deaktiviert, 1 = aktiviert)
5	extrahiert Moment 0 (niederen 16 Bits) (0 = deaktiviert, 1 = aktiviert)
4	extrahiert Threshold (0 = deaktiviert, 1 = aktiviert)
3	extrahiert maximale Intensität (0 = deaktiviert, 1 = aktiviert)
2	extrahiert Reflektionsbreite (0 = deaktiviert, 1 = aktiviert)
1	extrahiert Positions-Koordinate (X) (0 = deaktiviert, 1 = aktiviert)
0	extrahiert Abstands-Koordinate (Z) (0 = deaktiviert, 1 = aktiviert)

Jeder Punkt einer auszugebenden Eigenschaft (Abstands-Koordinate, Positions-Koordinate, ...) hat eine Auflösung von 16 Bit (2 Byte).

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
`ERROR_SETGETFUNCTIONS_REARRANGEMENT_PROFILE`	-159	Der `Rearrangement`-Parameter ist falsch

Die Breite eines Bildes zu errechnet sich folgendermaßen:

Breite = Anzahl der Eigenschaften * 2 * Resolution * Anzahl der Streifen

Der Timestamp kann in jeder Zeile in den letzten 16 Spalten eingeblendet werden. Soll er verwendet werden ist es sinnvoll ein leeres Feld für ihn einzufügen („Einfügen eines leeren Feldes“). Wird ein zusätzliches leeres Feld eingefügt, kommt zu der Breite noch 2 * Resolution hinzu.
Die Anzahlen der Eigenschaften und Streifen bezieht sich nur auf die auszugebenden und nicht auf die vorhandenen.

Nach dem Schreiben dieser Eigenschaft berechnet die LLT.dll selbständig die nötige Breite des Bildes und stellt sie ein. Die Breite kann wie in Kapitel 2.9.7 „Profile container size“ beschrieben ausgelesen werden.
Die Paketgröße zum Übertragen der umgewandelten Profile muss immer einem ganzzahligen vielfachen der Spaltenbreite entsprechen. Die LLT.dll stellt beim Starten der Übertragung automatisch eine passende Paketgröße, die maximal der aktuell eingestellten Paketgröße entspricht, ein.

Alternativ können auch aufeinander folgende Profile verbunden werden („Verbinden von aufeinander folgenden Profilen“), wobei dann die Bildgröße immer durch 16384 teilbar sein muss. Dies hat den Vorteil, dass größere Paketgrößen verwendet werden können und die Übertragung bei hohen Profilfrequenzen (> 2000 Hz) sicherer Funktioniert. Hierbei wird die aktuell eingestellte Paketgröße beibehalten und wenn nötig die Container-Höhe (bzw. die Anzahl der umgewandelten Profile pro Container) angepasst. Dies wird mit Hilfe des Rückgabewertes `GENERAL_FUNCTION_CONTAINER_MODE_HEIGHT_CHANGED` signalisiert. Die neue Höhe kann wie in Kapitel 2.9.7 „Profile container size“ beschrieben ausgelesen werden.
Die Beschränkungen für 512 und 1024 Punkte pro Profil können dem „OpManPartB.html“ entnommen werden.

Eine Beschreibung des Datenformates befindet sich in Kapitel 3.2 „Beschreibung des Datenformates CONTAINER“.

2.8.14. Profile filter

Mit Hilfe des Profil Filters können einfache Filter direkt im scanCONTROL auf ein Profil angewandt werden. Dadurch können z.B. ausreißende Punkte aussortiert werden.

Diese Funktion ist erst ab DSP-Version 17 vorhanden.

<table>
<tr><th>Bit</th><th>Funktion</th></tr>
<tr><td>11</td><td>Extrapoliert oder interpoliert ungültige Punkte. Erstellt neue gültige Punkte aus benachbarten gültigen Punkten während der ersten Filter Operation. Sollte nur im Zusammenhang mit Umskalierung verwendet werden. (0 = deaktiviert, 1 = aktiviert)</td></tr>
<tr><td>10</td><td>Umskalierung alle Informationen: wenn aktiviert werden alle Daten eines Streifens (Threshold, Reflektionsbreite, etc.) während der Skalierungs-Operation umskaliert. Sonst wird nur die Z-Koordinate umskaliert. (0 = deaktiviert, 1 = aktiviert)</td></tr>
<tr><td>9..7</td><td>Reserviert</td></tr>
<tr><td>6..4</td><td>Umskalierung des Profiles auf gleiche Abstände der Positions-Koordinate (äquidistante X-Abstände). Der Wertebereich liegt immer um das Zentrum x=0. Der Umskalierungsbereich sollte anhand des verwendeten Messfeldes ausgewählt werden, um over oder under sampling zu vermeiden. Abhängig vom Messbereich des scanCONTROL’s haben die Bereiche folgende Größen:
<table>
<tr><th>Bits 6..4</th><th>25mm</th><th>100mm</th></tr>
<tr><td>0</td><td>deaktiviert</td><td>deaktiviert</td></tr>
<tr><td>1</td><td>+/- 0.8mm</td><td>+/- 4.0mm</td></tr>
<tr><td>2</td><td>+/- 1.0mm</td><td>+/- 5.0mm</td></tr>
<tr><td>3</td><td>+/- 2.0mm</td><td>+/- 10mm</td></tr>
<tr><td>4</td><td>+/- 4.0mm</td><td>+/- 20mm</td></tr>
<tr><td>5</td><td>+/- 8.0mm</td><td>+/- 40mm</td></tr>
<tr><td>6</td><td>+/- 10mm</td><td>+/- 50mm</td></tr>
<tr><td>7</td><td>+/- 20mm</td><td>+/- 100mm</td></tr>
</table></td></tr>
<tr><td>3..2</td><td>Median Filter. Sollte nur in Verbindung mit Umskalierung auf äquidistante Positions-Koordinate verwendet werden. Die Filterbreite kann folgende Werte annehmen:
<table>
<tr><th>Bits 3..2</th><th>Filterbreite</th></tr>
<tr><td>0</td><td>deaktiviert</td></tr>
<tr><td>1</td><td>3</td></tr>
<tr><td>2</td><td>5</td></tr>
<tr><td>3</td><td>7</td></tr>
</table></td></tr>
<tr><td>1..0</td><td>Average Filter. Sollte nur in Verbindung mit Umskalierung auf äquidistante Positions-Koordinate verwendet werden. Die Filterbreite kann folgende Werte annehmen:
<table>
<tr><th>Bits 1..0</th><th>Filterbreite</th></tr>
<tr><td>0</td><td>deaktiviert</td></tr>
<tr><td>1</td><td>3</td></tr>
<tr><td>2</td><td>5</td></tr>
<tr><td>3</td><td>7</td></tr>
</table></td></tr>
</table>

2.9. Spezielle Eigenschafts-Funktionen

Mit Hilfe der speziellen Eigenschafts-Funktionen können weitere Eigenschaften des scanCONTROL's geschrieben und gelesen werden, welche nicht in das Konzept der normalen Eigenschafts-Funktionen passen.

2.9.1. Buffer count

```
int GetBufferCount(DWORD *pValue)
int SetBufferCount(DWORD Value)
```

Abfragen oder Setzen der Pufferanzahl im Scanner-Treiber. Je größer die Anzahl der Puffer ist, desto mehr Profile/Container können zwischengespeichert werden, bevor sie von der LLT.dll abgeholt werden müssen.
Diese Funktion steht nur bei einer Firewire-Übertragung zur Verfügung.

Eine hohe Pufferanzahl ist vor allem bei sehr hohen Profilfrequenzen, langsamen Rechnern und/oder Rechnern bei denen mehrere Programme im Hintergrund laufen sinnvoll. Ist die Pufferanzahl zu gering kommt es zu Ausfällen in der Profilübertragung, sodass die Profilfrequenz sinkt. Standardmäßig sind 20 Puffer eingestellt.
Soll eine Übertragung im Container-Mode gestartet werden ist darauf zu achten das die Anzahl der verwendeten Puffer maximal 4 beträgt. Sonst kann es zu einem sehr langsamen starten der Profilübertragung und eines sehr hohen Speicherverbrauches kommen. Werden sehr große Container verwendet können auch 3 Puffer.

Um Probleme bei der Puffersynchronisation bei externen Trigger zu vermeiden, sollte sichergestellt sein, dass innerhalb von 10 Sekunden mindestens so viele Triggerimpulse ausgelöst werden wie Puffer verwendet oder nur 2 Puffer verwendet werden.

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen"):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_SETGETFUNCTIONS_WRONG_BUFFER_COUNT	-150	Die Anzahl der gewünschten Puffer liegt nicht im Bereich >=2 und <= 200

2.9.2. Main reflection

```
int GetMainReflection(DWORD *pValue)
int SetMainReflection(DWORD Value)
```

Abfragen oder Setzen des auszugebenden Streifens bei einer Profilkonfiguration von `PURE_PROFILE` oder `QUADER_PROFILE`. Dieser Index wird in der LLT.dll zur Umwandlung von `PROFILE` in `PURE_PROFILE` und `QUADER_PROFILE` benötigt.
Der Index des auszugebenden Streifens geht von 0 für den 1. Streifen bis 3 für den 4. Streifen.
Eine Beschreibung der Profil-Daten und der Streifen befindet sich in Kapitel 3.1 „Beschreibung der Profil-Daten".

Diese Funktion steht nur bei einer Firewire-Übertragung zur Verfügung.

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_SETGETFUNCTIONS_REFLECTION_NUMBER_TOO_HIGH	-154	Der Index des auszugebenden Streifens ist größer 3

2.9.3. Max filesize

```
int GetMaxFileSize(DWORD *pValue)
int SetMaxFileSize(DWORD Value)
```

Abfragen oder Setzen der maximalen Dateigröße beim Speichern von Profilen in Byte. Ist diese Größe erreicht stoppt das Speichern.

2.9.4. Packet size

```
int GetMinMaxPacketSize(unsigned long *pMinPacketSize,
                        unsigned long *pMaxPacketSize)
```

Abfragen der minimalen und maximalen Paketgröße für die isochrone Profilübertragung auf dem Firewire-Bus.
Diese Funktion steht nur bei einer Firewire-Übertragung zur Verfügung.

```
int GetPacketSize(DWORD *pValue)
int SetPacketSize(DWORD Value)
```

Abfragen oder Setzen der aktuellen Paketgröße für die isochrone Profilübertragung auf dem Firewire-Bus. Diese Paketgröße muss zwischen der minimalen und maximalen Paketgröße liegen. Vom scanCONTROL werden die Paketgrößen 128, 256, 512, 1024, 2048 und 4096 Bytes unterstützt.
Bei Paketgrößen unter 4096 Bytes kann es bei hohen Profilfrequenzen zu Fehlern auf dem Firewire-Bus kommen, die sich in der Übertragung von kaputten Profilen oder einem starken Absinken der Profilfrequenz äußern.
Sollen mehrere scanCONTROLs gleichzeitig an einem Firewire-Bus betrieben werden, muss die maximale Paketgröße von 4096 Bytes zwischen ihnen aufgeteilt werden.

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_SETGETFUNCTIONS_PACKET_SIZE	-151	Die gewünschte Paketgröße wird nicht unterstützt

```
unsigned long MaxPacketSize = 0;

if(pInterfaceLLT->GetMinMaxPacketSize(NULL, &MaxPacketSize) > 0)
{
  pInterfaceLLT->SetPacketSize(MaxPacketSize);
}
```

2.9.5. Profile config

```
int GetProfileConfig(TProfileConfig *pValue)
int SetProfileConfig(TprofileConfig Value)
```

Abfragen oder Setzen der Profilkonfiguration.

ProfileConfig	Wert	Beschreibung
PROFILE	1	Profildaten aller vier Streifen
PURE_PROFILE	2	Reduzierte Profildaten eines Streifens (nur Positions- und Abstands-Werte)
QUARTER_PROFILE	3	Profildaten eines Streifens
PARTIAL_PROFILE	5	Partielles Profile welches per SetPartialProfile eingeschränkt wurde
CONTAINER	1	Container-Daten
VIDEO_IMAGE	1	Video-Bild des scanCONTROL's

Bei einer Verbindung über die serielle Schnittstelle wird nur die Profilkonfiguration PURE_PROFILE unterstützt.
Die Profilkonfiguration gilt gleichzeitig für das Speichern von Profilen und den Callback. Weiterführend siehe Kapitel 3 „Profil-/Container/Video-Übertragung".

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen"):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_SETGETFUNCTIONS_WRONG_PROFILE_CONFIG	-152	Die gewünschte Profilkonfiguration steht nicht zur Verfügung

```
TProfileConfig ProfileConfig;

pInterfaceLLT->GetProfileConfig(&ProfileConfig);

if(ProfileConfig != PURE_PROFILE)
{
  ProfileConfig = PURE_PROFILE;
  pInterfaceLLT->SetProfileConfig(&ProfileConfig);
}
```

2.9.6. Resolution

```
int GetResolution(DWORD *pValue)
int SetResolution(DWORD Value)
```

Abfragen oder Setzen der Auflösung von Profilen. Jeder Scanner unterstützt standardmäßig 256, 512 und 1024 Punkte pro Profil.
Zusätzlich können noch weitere Auflösungen unterstützt werden.

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_SETGETFUNCTIONS_NOT_SUPPORTED_RESOLUTION	-153	Die gewünschte Auflösung wird nicht unterstützt

Alle möglichen Auflösungen können mit Hilfe der Funktion `GetResolutions` ausgelesen werden.

```
int GetResolutions(DWORD *pValue, unsigned int nSize)
```

Dieser Funktion muss ein Feld von DWORD-Variablen übergeben werden. In dieses Feld werden dann alle möglichen Auflösungen eingetragen. Zurzeit sind nur maximal 6 verschiedene Auflösungen möglich. Ist die Größe des Feldes zu klein wird `ERROR_GETRESOLUTIONS_SIZE_TO_LOW` zurückgegeben, sonst die Anzahl der eingetragenen Auflösungen.

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_SETGETFUNCTIONS_SIZE_TOO_LOW	-156	Die Größe des übergebenen Feldes ist zu klein

Je größer die Auflösung der Profile ist, desto geringer ist die maximale Profilfrequenz (siehe Kapitel „Maximum Frequencies of Profile Measurements“ in der „QuickReference.html“).

Die Auflösung kann nur dann geändert werden, wenn keine Profile übertragen werden. Außerdem werden bei `SetResolution` alle Einstellungen für das `PartialProfile` gelöscht (siehe Kapitel 2.13 „PartialProfile-Funktionen“).

2.9.7. Profile container size

```
int GetMaxProfileContainerSize(unsigned int *pMaxWidth,
                               unsigned int *pMaxHeight)
```

Abfragen der maximalen `ProfileContainerSize` für den Container-Mode. (siehe Kapitel 2.8.13 „Rearrangement profile“).
Ist die maximale Breite 64, so wird der Container-Mode nicht von dem scanCONTROL unterstützt.

```
int GetProfileContainerSize(unsigned int *pWidth,
                            unsigned int *pHeight)
int SetProfileContainerSize(unsigned int nWidth,
                            unsigned int nHeight)
```

Abfragen oder Setzen der Größe des Containers für den Container-Mode.
Die Breite wird automatisch beim Aufruf von `SetFeature(FEATURE_FUNCTION_REARRANGEMENT_PROFILE)` gesetzt (siehe Kapitel 2.8.13 „Rearrangement profile“).
Die Höhe kann frei zwischen 0 und der maximal möglichen Höhe gewählt werden und entspricht der Anzahl von Profilen in dem Container übertragen werden.

Die Container-Höhe darf nicht höher als die dreifache Profilrate (siehe Kapitel 2.8.5 „Shutter time“) sein, da sichergestellt sein muss, dass mindestens alle 3 Sekunden ein Container übertragen wird. Sonst kann es zu erheblichen ausfällen kommen.

Werden nicht alle Parameter benötigt, kann für die nicht benötigten alternativ auch `NULL` übergeben werden.

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
`ERROR_SETGETFUNCTIONS_WRONG_PROFILE_SIZE`	-157	Die Größe für den Container ist falsch
`ERROR_SETGETFUNCTIONS_MOD_4`	-158	Die Container-Breite ist nicht durch 4 Teilbar

Ist „Verbinden von aufeinanderfolgenden Profilen“ aktiviert (siehe Kapitel 2.8.13 „Rearrangement profile“) muss die Höhe * Breite eines Bildes ein ganzzahliges vielfaches von 16384 sein. Wird versucht ein anderen Höhenwert einzustellen wird die Höhe automatisch auf den nächsten passenden Wert gesetzt. Zusätzlich wird der Fehlerwert `GENERAL_FUNCTION_CONTAINER_MODE_HEIGHT_CHANGED` ausgegeben um auf die Änderung aufmerksam zu machen.

2.10. Register-Funktionen

Funktionen zum Registrieren eines Callbacks für Profile und einer Message für Fehlermeldungen.

2.10.1. Callback

```
int RegisterCallback(TCallbackType CallbackType,
                     void *pLLTProfileCallback, void *pUserData)
```

Registrieren eines Callbacks vom Typ `TNewProfile_s` oder `TNewProfile_c` welcher immer dann aufgerufen wird wenn ein neues Profil/Container empfangen wurde. Dieser Callback muss nur dann registriert werden, wenn er verwendet werden soll.
Mit Hilfe des Parameters `CallbackType` wird die Aufrufkonvention des Callbacks festgelegt. Stimmt die Aufrufkonvention nicht mit der Aufrufkonvention der eigenen Callback-Funktion überein kann es zu Programmabstürzen kommen.

CallbackType	Wert	Beschreibung
`STD_CALL`	0	Der Callback arbeitet mit `stdcall` (`TNewProfile_s`)
`C_DECL`	1	Der Callback arbeitet mit `cdecl` (`TNewProfile_c`)

Der Parameter `pUserData` dient zur Unterscheidung von welcher scanCONTROL-Instanz der Callback aufgerufen wurde. Es kann z.B. der Pointer zu einer Klasse oder eine Nummer übergeben werden.
Wird dieser Funktion `NULL` als `pLLTProfileCallback` übergeben, wird der Callback wieder deaktiviert. Wobei darauf zu achten ist, das die Aufrufkonvention auch hier korrekt mit angegeben werden muss.

Weitere Informationen zu diesem Thema sind in den Kapiteln 2.11 „Profilübertragungs-Funktionen“ und 3 „Profil-/Container/Video-Übertragung“ zu finden.

```
typedef void (_stdcall *TNewProfile_s)(const unsigned char *pData,
                                       unsigned int nSize, void *pUserData);
typedef void (_cdecl *TNewProfile_c)(const unsigned char *pData,
                                     unsigned int nSize, void *pUserData);
```

Diese Callbacks werden, wenn sie registriert wurden, nach dem Empfangen eines Profils/Containers aufgerufen und besitzt als Parameter einen Pointer auf die Profil-/Container-Daten, die dazugehörige Größe des Datenfeldes und einen `pUserData`-Parameter. Mit dem `pUserData`-Parameter kann zum Beispiel unterschieden werden von welchem scanCONTROL das Profil / der Container stammt.

Der Callback ist für die Verarbeitung von Profilen/Containern mit einer hohen Profilfrequenz gedacht. Innerhalb des Callback können die Profile/Container in einen Puffer für eine spätere oder zum Callback synchrone oder asynchrone Verarbeitung kopiert werden. Eine Verarbeitung innerhalb des Callbacks ist nicht zu empfehlen, da für die Zeit die der Callback zur Verarbeitung benötigt die LLT.dll keine neuen Profile/Container vom Treiber abholen kann. Unter Umständen kann es dadurch zu Profil-/Container-Ausfällen kommen.
Die Profil-/Container-Daten in dem vom Callback übergebenen Puffer dürfen nicht verändert werden.

Die Profilkonfiguration des Pointers kann über die Funktion `SetProfileConfig` eingestellt werden und gilt gleichzeitig auch für das Speichern von Profilen/Containern (siehe Kapitel 2.9.5 „Profile config“).

```
#include <vector>
HANDLE hProfileEvent;
//Anlegen eines Puffers fuer ein Profil mit der maximalen Profilgroesse
std::vector<unsigned char> ProfileBuffer(1024*64);

void GetProfileFromCallback()
{
  //Erstellen eines Events
  hProfileEvent = CreateEvent(NULL, true, false, "ProfileEvent");

  pInterfaceLLT->RegisterCallback(STD_CALL, (void*)NewProfile_s, NULL);

  //Aktivieren der Profiluebertragung
  if(pInterfaceLLT->TransferProfiles(NORMAL_TRANSFER, true)
                                          <= GENERAL_FUNCTION_NOT_AVAILABLE)
    return;

  //Warten auf ein Event vom Profil-Callback
  if(WaitForSingleObject(hProfileEvent, 1000) != WAIT_OBJECT_0)
  {
    //Fehler beim warten auf den Callback
    return;
  }
  //Auswerten des Profiles

  //Deaktivieren der Profiluebertragung
  if(pInterfaceLLT->TransferProfiles(NORMAL_TRANSFER, false)
                                          <= GENERAL_FUNCTION_NOT_AVAILABLE)
    return;
}
void CALLBACK NewProfile_s(const unsigned char *pData, unsigned int nSize,
                           void *pUserData)
{
  if(ProfileBuffer.size() >= nSize)
  {
    //Wenn die Profil kleiner gleich der Puffergroesse ist:
    //Kopieren des Profiles in den Puffer
    memcpy(&ProfileBuffer[0], pData, nSize);
    SetEvent(hProfileEvent);
  }
}
```

2.10.2. Message

```
int RegisterErrorMsg(UINT Msg, HWND hWnd, WPARAM WParam)
```

Registrieren einer Fehler-Message welche bei Fehlern gesendet wird.

Fehlercode im LPARAM	Wert	Beschreibung
ERROR_SERIAL_COMM	1	Fehler während der seriellen Datenübertragung. Eventuell ist die Profilfrequenz zu hoch.
ERROR_SERIAL_LLT	7	scanCONTROL konnte Kommando nicht verstehen oder es wurde ein Parameter außerhalb des Gültigkeitsbereiches gesendet.
ERROR_CONNECTIONLOST	10	Die Verbindung zum Scanner wurde unterbrochen (Scanner wurde Abgeschaltet, reseted oder das Firewire-Kabel wurde entfernt). Bitte senden sie ein „Disconnect“ aus um sich neu verbinden zu können. Diese Message wird nur bei einer Verbindung über Firewire gesendet.
ERROR_STOPSAVING	100	Das Speichern von Profilen ist beendet (maximale Dateigröße erreicht).

2.11. Profilübertragungs-Funktionen

Beschreibung der Funktionen für die Profilübertragung. Genauer wird darauf im Kapitel 3 „Profil-/Container/Video-Übertragung“ eingegangen.

Beispiele zum Übertragen von Profilen mit unterschiedlichen Profilkonfigurationen und das Umrechnen der Profildaten in Millimeter sind im Kapitel 4 „LLT2800Samples“ zu finden.

2.11.1. Transfer profiles

```
int TransferProfiles(int TransferProfileType, int nEnable);
```

Funktion zum Starten oder Beenden der Profilübertragung.
Der Parameter `TransferProfileType` gibt an, ob eine kontinuierliche oder eine Bedarfsmäßige Übertragung aktiviert werden soll.

TransferProfileType	Wert	Beschreibung
NORMAL_TRANSFER	0	Aktivieren einer kontinuierlichen Übertragung von Profilen
SHOT_TRANSFER	1	Aktivieren einer Bedarfsmäßigen Übertragung von Profilen (die Übertragung wird immer per `MultiShot` aktiviert)
NORMAL_CONTAINER_MODE	2	Aktivieren einer kontinuierlichen Übertragung im Container-Mode
SHOT_CONTAINER_MODE	3	Aktivieren einer Bedarfsmäßigen Übertragung im Container-Mode (die Übertragung wird immer per `MultiShot` aktiviert)

Der Parameter `nEnable` gibt an, ob eine Übertragung gestartet oder beendet werden soll. Nach dem starten einer Übertragung kann es bis zu 100 ms dauern ehe die ersten Profile/Container per Callback ankommen oder per `GetActualProfile` abgeholt werden können.

Wird eine Übertragung beendet, wartet die Funktion automatisch, bis der Treiber alle Puffer zurückgegeben hat. Dies kann bis zu 20 Sekunden dauern.

Soll eine Übertragung im Container-Mode gestartet werden (siehe Kapitel 2.8.13 „Rearrangement profile“) ist darauf zu achten das die Anzahl der verwendeten Puffer maximal 4 beträgt (siehe Kapitel 2.9.1 „Buffer count“). Sonst kann es zu einem sehr langsamen starten der Profilübertragung und eines sehr hohen Speicherverbrauches kommen. Werden sehr große Container verwendet, reichen auch 3 Puffer.
Zudem ist zu beachten, dass spätestens aller 3 Sekunden ein Container übertragen werden muss (siehe Kapitel 2.8.5 „Shutter time“ und Kapitel 2.9.7 „Profile container size“). Sonst kann es zu erheblichen ausfällen kommen.

Der Rückgabewert ist die Größe eines Profiles/Containers. Ist die Größe kleiner oder gleich `GENERAL_FUNCTION_NOT_AVAILABLE` ist es einer der allgemeinen oder der folgenden Rückgabewerte (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
`ERROR_PROFTRANS_PACKET_SIZE_TOO_HIGH`	-107	Die Paketgröße ist größer als die verfügbare -> mit `SetPacketSize` eine niedrigere Paketgröße einstellen
`ERROR_PROFTRANS_CREATE_BUFFERS`	-108	Die Puffer für den Treiber konnten nicht ordnungsgemäß angelegt werden -> ev. PC neu starten

2.11.2. Transfer video stream

```
int TransferVideoStream(int TransferVideoType, int nEnable,
                        unsigned int *pWidth, unsigned int *pHeight)
```

Funktion zum Starten oder Beenden der Übertragung von Video-Bildern des Bildsensors.
Der Parameter `TransferVideoType` gibt den verwendeten Übertragungsmode an.

TransferVideoType	Wert	Beschreibung
`VIDEO_MODE_0`	0	Verkleinertes Bild der Matrix
`VIDEO_MODE_1`	1	Vollständiges Bild der Matrix

Die Größe der übertragenen Bilder von der Matrix werden in die Parametern `pWidth` und `pHeight` kopiert.

Zu beachten ist, dass die Video-Bilder nur mit maximal 25 Bildern pro Sekunde übertragen werden können. Die Werte für die `Shutter time` und `Idle time` müssen dem entsprechend geändert werden (siehe Kapitel 2.8.5 „Shutter time“ und Kapitel 2.8.6 „Idle time“).
Video-Bilder können nur mit `GetActualProfile` abgeholt. Per `Callback` stehen sie nicht zur Verfügung.
Video Bilder können nur einzeln als Bitmap gespeichert werden (siehe Kapitel 2.16.1. „Speichern von Profilen“).

Der Rückgabewert ist die Größe eines Video Bildes. Ist die Größe kleiner oder gleich GENERAL_FUNCTION_NOT_AVAILABLE ist es einer der allgemeinen oder der folgenden Rückgabewerte (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_PROFTRANS_PACKET_SIZE _TOO_HIGH	-107	Die Paketgröße ist größer als die verfügbare -> mit SetPacketSize eine niedrigere Paketgröße einstellen
ERROR_PROFTRANS_CREATE_BUFFERS	-108	Die Puffer für den Treiber konnten nicht ordnungsgemäß angelegt werden -> ev. PC neu starten

2.11.3. Multi shot

Mit Hilfe der Funktionen MultiShot ist es möglich nur ein Profil/Container oder eine bestimmte Menge an Profilen/Containern zu übertragen.
Dieses Feature steht erst ab einer DSP-Firmwareversion 10 zur Verfügung.

```
int MultiShot(unsigned int nCount)
```

Anfordern von mehreren Profilen/Containern. Die Anzahl der Profile/Container wird in dem Parameter Count übergeben. Es können zwischen 1 und 65535 Profile/Container angefordert werden.
Dafür muss beim starten der Übertragung der SHOT_TRANSFER- oder der SHOT_CONTAINER_MODE-Mode ausgewählt worden sein.

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_PROFTRANS_SHOTS_NOT _ACTIVE	-100	Der SHOT_TRANSFER-Mode oder der SHOT_CONTAINER_MODE-Mode ist nicht aktiviert -> Profilübertragung neu starten
ERROR_PROFTRANS_SHOTS_COUNT _TOO_HIGH	-101	Die Anzahl der angeforderten Profile/Container ist größer als 65535

Um Probleme bei der Puffersynchronisation zu vermeiden, sollten Sie Sicherstellen das innerhalb von 10 Sekunden mindestens so viele Profile/Container wie es Puffer gibt übertragen werden oder nur 2 Puffer verwendet werden (siehe Kapitel 2.9.1 „Buffer count“).

Durch deaktivieren der Profilübertragung nach dem empfangen aller angeforderten Profile können Probleme bei der Puffersynchronisation ebenfalls vermieden werden.

2.11.4. Get profile

```
int GetProfile();
```

Übertragen eines Profiles über die serielle Schnittstelle. Diese Funktion gibt es nur für die serielle Schnittstelle.

2.11.5. Get actual profile

```
int GetActualProfile(unsigned char *pBuffer, unsigned int nBuffersize,
                     TProfileConfig ProfileConfig, unsigned int *pLostProfiles)
```

Abholen des aktuellen Profils/Containers/Video-Bildes. Dafür muss der Funktion ein Puffer übergeben werden, in den das Profil, der Container oder das Video-Bild kopiert wird. Über den Parameter `ProfileConfig` kann die für diese Abfrage gewünschte Profilkonfiguration (siehe Kapitel 3 „Profil-/Container/Video-Übertragung“ und Kapitel 2.9.5 „Profile config“) ausgewählt werden. Ist der Container-Mode aktiviert können die Container nur mit `PROFILE` abgeholt werden.

Mit Hilfe des Pointers `pLostProfiles` kann die Anzahl der verloren gegangenen Profile/Container abgefragt werden. Dieser Wert ist größer 0, wenn zwischen zwei Aufrufen der Funktion mehrere Profile/Container empfangen wurden. Wird dieser Parameter nicht benötigt, kann alternativ auch `NULL` übergeben werden.

Der Rückgabewert ist die Anzahl der in den Puffer kopierten Bytes. Ist die Anzahl kleiner oder gleich `GENERAL_FUNCTION_NOT_AVAILABLE` ist es einer der allgemeinen oder der folgenden Rückgabewerte (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
`ERROR_PROFTRANS_WRONG_PROFILE_CONFIG`	-102	Kann das geladen Profil nicht in die gewünschte Profilkonfiguration konvertieren
`ERROR_PROFTRANS_FILE_EOF`	-103	Das Dateiende beim Laden von Profilen ist erreicht
`ERROR_PROFTRANS_NO_NEW_PROFILE`	-104	Es ist seit dem letzten Aufruf von `GetActualProfile` kein neues Profil angekommen
`ERROR_PROFTRANS_BUFFER_SIZE_TOO_LOW`	-105	Die Puffergröße des übergebenen Puffers ist zu klein
`ERROR_PROFTRANS_NO_PROFILE_TRANSFERE`	-106	Die Profilübertragung ist nicht gestartet und es wird keine Datei geladen

Näheres zur Profilübertragung im Kapitel 3 „Profil-/Container/Video-Übertragung“.

2.11.6. Konvertieren von Profil-Daten

```
int ConvertProfile2Values(const unsigned char *pProfile,
    unsigned int nResolution, TProfileConfig ProfileConfig,
    TScannerType ScannerType, unsigned int nReflection, int bConvertToMM,
    unsigned short *pWidth, unsigned short *pMaximum,
    unsigned short *pThreshold, double *pX, double *pZ,
    unsigned int *pM0, unsigned int *pM1)

int ConvertPartProfile2Values(const unsigned char *pProfile,
    TPartialProfile *pPartialProfile, TScannerType ScannerType,
    unsigned int nReflection, int bConvertToMM,
    unsigned short *pWidth, unsigned short *pMaximum,
    unsigned short *pThreshold, double *pX, double *pZ,
    unsigned int *pM0, unsigned int *pM1);
```

Konvertieren von Profilen oder partiellen Profilen in Millimeter-Werte für die Positions-Koordinate und die Abstands-Koordinate. Außerdem können alle anderen enthaltenen Informationen extrahiert werden.
Es gibt zwei Versionen der Funktion, eine für normale Profile und eine für das `PARTIAL_PROFILE`-Format.

Parameter	**Beschreibung**
`const unsigned char *pProfile`	Pointer zu dem Profil
`unsigned int nResolution`	Punkte pro Profil (64 ... 1024)
`TProfileConfig ProfileConfig`	Profil Konfiguration (`PURE_PROFILE` ... `PROFILE`)
`TScannerType ScannerType`	Messbereich des scanCONTROLs (`LLT25` oder `LLT100`)
`unsigned int nReflection`	Auszulesender Streifen (0 bis 3)
`int bConvertToMM`	Konvertieren der Positions- und Abstands-Koordinaten in Millimeter (0 = deaktiviert, 1 = aktiviert)
`TPartialProfile *pPartialProfile`	PartialProfile
`unsigned short *pWidth`	Pointer zu einem Array für die Reflektionsbreiten
`unsigned short *pMaximum`	Pointer zu einem Array für die maximalen Intensitäten
`unsigned short *pThreshold`	Pointer zu einem Array für die Thresholds
`double *pX`	Pointer zu einem Array für die Positions-Koordinaten
`double *pZ`	Pointer zu einem Array für die Abstands-Koordinaten
`unsigned int *pM0`	Pointer zu einem Array für die M0s
`unsigned int *pM1`	Pointer zu einem Array für die M1s

Die Arrays müssen mindestens die Größe der Auflösung (Punkte pro Profile) bzw. des `PointCounts` bei `PARTIAL_PROFILE` besitzen.
Werden nicht alle Informationen benötigt, können für die nicht benötigten Informationen anstatt des Arrays eine `NULL` übergeben werden.

Die Funktionen führen automatisch einen Test durch, welche Informationen in den jeweiligen Profilen vorhanden sind.

Ist der Rückgabewert größer 0 war die Funktion erfolgreich und in den einzelnen Bits des Rückgabewertes sind die gültigen Arrays codiert.

Gesetztes Bit	Konstante für das Bit	Beschreibung
8	CONVERT_WIDTH	Das Array für die Reflektionsbreite wurde mit Daten gefüllt
9	CONVERT_MAXIMUM	Das Array für die maximalen Intensitäten wurde mit Daten gefüllt
10	CONVERT_THRESHOLD	Das Array für die Thresholds wurde mit Daten gefüllt
11	CONVERT_X	Das Array für die Positions-Koordinaten wurde mit Daten gefüllt
12	CONVERT_Z	Das Array für die Abstands-Koordinaten wurde mit Daten gefüllt
13	CONVERT_M0	Das Array für die M0s wurde mit Daten gefüllt
14	CONVERT_M1	Das Array für die M1s wurde mit Daten gefüllt

Ist die Anzahl kleiner oder gleich GENERAL_FUNCTION_NOT_AVAILABLE ist es einer der allgemeinen oder der folgenden Rückgabewerte (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_PROFTRANS_REFLECTION_NUMBER_TOO_HIGH	-110	Die Nummer der gewünschten Streifens ist größer 3

Für weitere Informationen siehe Kapitel 3 „Profil-/Container/Video-Übertragung“.

2.12. Is-Funktionen

Funktionen zum Abfragen von Zuständen und Verbindungen.

```
int IsSerial()
```

Ist der Scanner über die serielle Schnittstelle verbunden?

```
int IsFirewire()
```

Ist der Scanner über die Firewire verbunden?

```
int IsTransferingProfiles()
```

Überträgt der Scanner momentan Daten?

Der Rückgabewert kann einer der folgenden Rückgabewerte sein:

Konstante für den Rückgabewert	Wert	Beschreibung
IS_FUNC_YES	1	Abgefragter Zustand oder Verbindung ist aktiv
IS_FUNC_NO	0	Abgefragter Zustand oder Verbindung ist nicht aktiv

2.13. PartialProfile-Funktionen

Das scanCONTROL bietet die Möglichkeit das übertragene Profil einzuschränken. Der Vorteil von diesem Verfahren ist eine geringere Größe der übertragenen Daten. Außerdem können damit nicht benötigte Bereiche eines Profils schon direkt im scanCONTROL weg geschnitten werden.
Dieses Feature steht erst ab einer DSP-Firmwareversion von 13 zur Verfügung und kann nicht mit dem Container-Mode kombiniert werden.

Beispiele zum Übertragen von Profilen mit „Partial Profile“ sind im Kapitel 4 „LLT2800Samples“ zu finden.

2.13.1. GetPartialProfileUnitSize

```
int GetPartialProfileUnitSize(unsigned int *pUnitSizePoint,
                              unsigned int *pUnitSizePointData)
```

Diese Funktion gibt die Schrittweiten zum Einstellen des partiellen Profils zurück. Ist der Parameter `pUnitSizePoint` gleich der Resolution des uneingeschränkten Profils wird dieses Feature noch nicht von der Firmware-Version Ihres scanCONTROL’s unterstützt.

2.13.2. Get/SetPartialProfile

```
int GetPartialProfile(TPartialProfile *pPartialProfile)
int SetPartialProfile(TPartialProfile *pPartialProfile)
```

Mit Hilfe dieser Funktion kann die partielle Profilübertragung des scanCONTROL’s eingestellt werden. Vorher muss die Profile Konfiguration auf `PARTIAL_PROFILE` gestellt werden (siehe Kapitel 2.9.5 „Profile config“).

Parameter	**Bedeutung**
`nStartPoint`	Nummer des ersten zu übertragenden Punktes
`nStartPointData`	Erstes Byte der Punkte
`nPointCount`	Anzahl der zu übertragenden Punkte
`nPointDataWidth`	Anzahl der Bytes pro Punkt

Alle Parameter der `SetPartialProfile` Funktion müssen immer ein Vielfaches der jeweiligen `nUnitSize` der Funktion `GetPartialProfileUnitSize` sein.

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_PARTPROFILE_NO_PART_PROF	-350	Die Profilkonfiguration ist nicht auf PARTIAL_PROFILE eingestellt -> SetProfileConfig(PARTIAL_PROFILE); aufrufen
ERROR_PARTPROFILE_TOO_MUCH_BYTES	-351	Die Anzahl der Bytes pro Punkt ist zu hoch -> nStartPointData oder nPointDataWidth ändern
ERROR_PARTPROFILE_TOO_MUCH_POINTS	-352	Die Anzahl der Punkte ist zu hoch -> nStartPoint oder nPointCount ändern
ERROR_PARTPROFILE_NO_POINT_COUNT	-353	nPointCount oder nPointDataWidth ist 0
ERROR_PARTPROFILE_NOT_MOD_UNITSIZE_POINT	-354	nStartPoint oder nPointCount sind kein vielfaches von nUnitSizePoint (siehe Kapitel GetPartialProfileUnitSize 2.13.1 „GetPartialProfileUnitSize“)
ERROR_PARTPROFILE_NOT_MOD_UNITSIZE_DATA	-355	nStartPointData oder nPointDataWidth sind kein vielfaches von nUnitSizePointData (siehe Kapitel GetPartialProfileUnitSize 2.13.1 „GetPartialProfileUnitSize“)

Der Aufbau eines Profiles ist in Kapitel 3.1 „Beschreibung der Profil-Daten“ beschrieben.
Im Normalfall werden immer nur die Position (X-Wert) und der Abstand (Z-Wert) des ersten Streifens benötigt.

2.14. Time-Funkionen

```
void Timestamp2TimeAndCount(const unsigned char *pTimestamp,
                double *pTimeShutterOpen, double *pTimeShutterClose,
                unsigned int *pProfileCount)
```

Diese Funktion wertet den gesamten Timestamp eines Profils aus. Sie gibt den Timestamp des Anfangs und des Endes der Belichtung und die fortlaufende Profilnummer zurück.

```
void Timestamp2CmmTriggerAndInCounter(const unsigned char *pTimestamp,
                            unsigned int *pInCounter, bool *pCmmTrigger,
                            bool *pCmmActive, unsigned int *pCmmCount)
```

Diese Funktion wertet nur den optionalen Teil des Timestamps eines Profils aus. Sie gibt den Zählerstand des internen Zählers, die CmmTrigger und CmmActive Flags sowie den CMM-Trigger-Zähler zurück.

Das `CmmTrigger` Flag ist immer in dem Profil gesetzt in welchem auch ein CMM Triggerimpuls ausgegeben wurde. Das `CmmActive` Flag ist immer gesetzt wenn der CMM-Trigger im Allgemeinen aktiviert ist.
Für nicht benötigte Parameter kann alternativ auch `NULL` übergeben werden.

```
#include <vector>

DWORD Resolution = 0;
double ShutterOpen, ShutterClose;
unsigned int ProfileCount, InCounter, CmmCount;
bool CmmTrigger, CmmActive;

//Abfragen der Aufloesung
pInterfaceLLT->GetResolution(&Resolution);

//Erstellen eines Puffers fuer die Profile
std::vector<unsigned char> ProfilData((Resolution * 4) + 16);

if(pInterfaceLLT->GetActualProfile
   (&ProfilData[0], ProfileData.size(), PURE_PROFILE, NULL) ==
  ProfileData.size())
{
  //Dekodieren des Zeitstempels und des Profil-Zaehlers
  pInterfaceLLT->Timestamp2TimeAndCount(&ProfilData[(Resolution * 4)],
        &ShutterOpen, &ShutterClose, ProfileCount);

  //Dekodieren des optionalen Zaehlers
  pInterfaceLLT->Timestamp2CmmTriggerAndInCounter(
        &ProfilData[(Resolution * 4)], &InCounter, &CmmTrigger, &CmmActive,
        &CmmCount);
}
```

2.15. Postprocessing-Funktionen

Durch das Postprocessing kann das scanCONTROL mehrere Module auf die Profile anwenden. Diese Module stehen nur in besonderen Optionen des scanCONTROL's zur Verfügung. Nähere Informationen entnehmen Sie bitte der entsprechenden Dokumentation.

2.15.1. Read/Write Postprocessing Parameter

```
int ReadPostProcessingParameter(DWORD *pParameter, unsigned int nSize)
```

Lesen der aktuellen Postprocessing-Parameter. Diese Parameter können maximal 250 DWORD’s lang sein.

```
int WritePostProcessingParameter(DWORD *pParameter, unsigned int nSize)
```

Schreiben der Postprocessing-Parameter. Diese Parameter können maximal 250 DWORD’s lang sein.

Der Rückgabewert kann einer der allgemeinen Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“).
Das Postprocessing kann über die „Processing profile data“-Eigenschaft aktiviert werden (siehe hierzu Kapitel 2.8.7 „Processing profile data“).

2.15.2. Postprocessing Results

Einige der Postprocessing-Module können Ergebnisse ihrer Berechnungen in das übertragene Profil integrieren. Sie werden auf die Positions- und Abstands-Koordinaten eines Streifens geschrieben.
Die folgende Funktion extrahiert aus einem gegebenen Profil die Rechenergebnisse:

```
int ConvertProfile2ModuleResult(const unsigned char *pProfileBuffer,
        unsigned int nProfileBufferSize,
        unsigned char *pModuleResultBuffer, unsigned int nResultBufferSize,
        TPartialProfile *pPartialProfile)
```

Der Funktion wird der Profilpuffer, seine Größe, einen Puffer in den das Ergebnis kopiert wird und dessen Größe übergeben. Wird ein Profil aus einer Datei geladen muss noch ein Pointer auf die aktuelle `TPartialProfile` Struktur übergeben werden.

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	**Wert**	**Beschreibung**
`ERROR_POSTPROCESSING_NO_PROF_BUFFER`	-200	Es wurde kein Profilpuffer übergeben
`ERROR_POSTPROCESSING_MOD_4`	-201	Der Parameter `nStartPointData` oder `nPointDataWidth` ist nicht durch 4 Teilbar
`ERROR_POSTPROCESSING_NO_RESULT`	-202	Kein Ergebnisblock im Profil gefunden
`ERROR_POSTPROCESSING_LOW_BUFFERSIZE`	-203	Die Puffergröße für das Ergebnis ist zu klein
`ERROR_POSTPROCESSING_WRONG_RESULT_SIZE`	-204	Die Größe des Ergebnisblocks im Profil ist nicht korrekt

2.16. File-Funktionen

Funktionen zum Laden und Speichern von Profilen oder Profilströmen. Profile können mit dem `AVI`-Datenformat (Standardisiertes Video-Format) oder dem `LLT`-Datenformat (Datenformat der LLT.dll) gespeichert und geladen werden.

Das `LLT`-Datenformat zum Abspeichern von Profilen ist in Kapitel 3.5 „Beschreibung der *.llt Dateien“ beschrieben.

Beispiele zum Laden und Speichern von Profilen sind im Kapitel 4 „LLT2800Samples“ zu finden.

2.16.1. Speichern von Profilen

```
int SaveProfiles(const char *pFilename, TFileType FileType)
```

Speichern von Profilen. Die Profile werden dabei mit der aktuellen Profilkonfiguration gespeichert. Der Dateiname muss inklusive Endung angegeben werden. Mit dem `FileType` kann der gewünschte Dateityp angegeben werden.

FileType	Wert	Beschreibung
`AVI`	0	AVI-Datei
`LLT`	1	LLT- Datei
`CSV`	2	CSV- Datei (nur für Profile)
`BMP`	3	BMP- Datei (nur für Video Bilder)

Es wird empfohlen das `AVI`-Datenformat zu verwenden, da dieses Format von allen Programmen für das scanCONTROL der Micro-Epsilon gelesen werden kann. Das `LLT`-Datenformat ist nur bei Anwendungen mit gleichzeitigem Speichern und versetztem Lesen nötig.
Zum Beenden des Speicherns muss `SaveProfiles(NULL)` aufgerufen werden. Wird beim Speichern die maximale Dateigröße erreicht wird eine Fehler-Message mit dem `ERROR_STOPSAVING` Wert gesendet (siehe Kapitel 2.10.2 „Message“).

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
`ERROR_LOADSAVE_WRITING_LAST _BUFFER`	-50	Fehler beim deaktivieren des Speicherns, die letzten Profile der Datei können beschädigt sein oder es wurden nicht alle gespeichert
`ERROR_LOADSAVE_AVI_NOT _SUPPORTED`	-58	Das Betriebssystem unterstützt das AVI-Format nicht, bitte benutzen Sie Windows 2000 oder höher
`ERROR_LOADSAVE_WRONG_PROFILE _CONFIG`	-60	Die Profilkonfiguration oder der Filetype passt nicht zu den übertragenen Profilen/Containern/Video-Bildern
`ERROR_LOADSAVE_NOT _TRANSFERING`	-61	Die Profilübertragung ist nicht aktiv

2.16.2. Laden von Profilen

```
int LoadProfiles(const char *pFilename, TPartialProfile *pPartialProfile,
        TProfileConfig *pProfileConfig, TScannerType *pScannerType,
        DWORD *pRearrengementProfile)
```

Laden von Profilen aus einer Datei. Dabei müssen fünf Pointer auf Variablen für den Dateinamen, die PartialProfile-Konfiguration, die Profilkonfiguration, den Scanner-Typ (Messbereich) und die `Rearrengement`-Eigenschaft übergeben werden. Die Pointer für die Profilkonfiguration und den Scanner-Typ können `NULL` sein.
Es können `*.LLT` und `*.AVI` Dateien geladen werden, welche mit der LLT.dll, dem LLT2800Demo Programm oder den scanCONTROL Programmen der Micro-Epsilon gespeichert wurden.

Nach dem Laden können mit der Funktion `GetActualProfile` die einzelnen Profile in der Datei nacheinander ausgelesen werden (siehe Kapitel 2.11.5 „Get actual profile“).
Dabei ist zu beachten, dass aus dem `PartialProfile`-Parameter die Größe für den Puffer für ein Profil/Container berechnet werden kann (`PointCount` * `PointDataWidth`).

Der Rückgabewert ist die Anzahl von Profilen/Containern in der geladenen Datei. Ist die Anzahl kleiner als `0` ist es einer der allgemeinen oder der folgenden Rückgabewerte (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	**Wert**	**Beschreibung**
`ERROR_LOADSAVE_WHILE_SAVE_PROFILE`	-51	Kann Datei nicht laden, da das Speichern aktiv ist
`ERROR_LOADSAVE_NO_PROFILELENGTH_POINTER`	-52	Es wurde kein Pointer für die Profillänge übergeben
`ERROR_LOADSAVE_NO_LOAD_PROFILE`	-53	Der Filename ist `NULL`, aber es wird momentan keine Datei geladen
`ERROR_LOADSAVE_STOP_ALREADY_LOAD`	-54	Es wurde schon eine Datei geladen, das laden wurde gestoppt
`ERROR_LOADSAVE_CANT_OPEN_FILE`	-55	Kann die Datei nicht öffnen
`ERROR_LOADSAVE_INVALID_FILE_HEADER`	-56	Der Fileheader der zu ladenden Datei ist falsch
`ERROR_LOADSAVE_AVI_NOT_SUPPORTED`	-58	Das Betriebssystem unterstützt das AVI-Format nicht. Bitte benutzen Sie Windows 2000 oder höher
`ERROR_LOADSAVE_NO_REARRANGEMENT_POINTER`	-59	Der Pointer `pRearrengementProfile` ist `NULL`

Die Profilkonfiguration der geladenen Profile sollte immer der Profilkonfiguration der `LoadProfiles`-Funktion entsprechen. Zusätzlich kann bei einer gespeicherten Profilkonfiguration von `PROFILE` auch `QUARTER_PROFILE` und `PURE_PROFILE` oder bei `QUARTER_PROFILE` auch `PURE_PROFILE` ausgelesen werden.

Das Laden einer Datei beeinflusst nicht den Profilkonfigurationswert für das Speichern von Profilen bzw. den Callback.

Zum Beenden des Ladens muss `LoadProfiles(NULL, NULL, NULL, NULL, NULL)` aufgerufen werden.

Mit einer zweiten Instanz der LLT.dll können über `LoadFile` Profile geladen werden, auch wenn in die gleiche Datei noch von der ersten Instanz gespeichert wird. Dieses gleichzeitige Speichern und Laden von Profilen ist nur mit dem `LLT`-Datenformat möglich.
Mit der Funktion `LoadProfilesGetPos` (siehe nächstes Kapitel) kann dabei die aktuelle Anzahl der Profile in der Datei abgefragt werden.

2.16.3. Navigieren in einer geladenen Datei

```
int LoadProfilesGetPos(unsigned int *pActualPosition,
                       unsigned int *pMaxPosition)
```

Mit dieser Funktion kann die Anzahl der Profile und die aktuelle Leseposition in der geladenen Datei abgefragt werden. Wird in die Datei noch von einer anderen Instanz gespeichert kann sich die Anzahl der Profile ändern.

```
int LoadProfilesSetPos(unsigned int nNewPosition)
```

Setzen der aktuellen Leseposition in einer geladenen Datei. Um die Position auf das erste Profil zu setzen muss `nNewPosition` 0 sein.

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
`ERROR_LOADSAVE_FILE_POSITION_TOO_HIGH`	-57	Die gewünschte Position ist größer oder gleich der maximalen Position

2.17. Spezielle CMM-Trigger-Funktionen

Mit Hilfe der speziellen CMM-Trigger-Funktionen wird das Starten und Beenden der Profilübertragung mit aktiviertem CMM-Trigger vereinfacht. Zusätzlich können die Profile mit aktivem CMM-Trigger in eine Datei gespeichert werden. Der CMM-Trigger steht nur in bestimmten Optionen des scanCONTROL’s zur Verfügung.
Zum CMM-Trigger siehe auch die Beschreibung in Kapitel 2.8.12 „CMMTrigger“.

Beispiele für den CMM-Trigger sind im Kapitel 4 „LLT2800Samples“ zu finden.

Starten der Profilübertragung mit CMM-Trigger:

```
int StartTransmissionAndCmmTrigger(DWORD cmmTrigger,
                    TTransferProfileType TransferProfileType,
                    unsigned int nProfilesForerun, const char *pFilename,
                    TFileType FileType, unsigned int nTimeout);
```

Beschreibung der Parameter:

Parameter	Beschreibung
`nCmmTrigger`	Erstes Befehlswort des CMM-Triggers, welches den Divisor und die Polarität enthält
`nProfilesForerun`	Anzahl der kontinuierlich eingegangenen Profile ab der eine stabile Datenübertragung angenommen wird
`pFilename`	Dateiname für die zu speichernde Datei (muss `NULL` sein, wenn kein speichern gewünscht)
`FileType`	Datenformat der zu speichernden Datei (siehe Kapitel 2.16.1 „Speichern von Profilen“)
`nTimeout`	Timeout in ms für die gesamte Funktion

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_CMMTRIGGER_NO_DIVISOR	-400	Divisor muss > 0 sein
ERROR_CMMTRIGGER_TIMEOUT _AFTER_TRANSFERPROFILES	-401	Es wurden nach TransferProfiles keine Profile empfangen
ERROR_CMMTRIGGER_TIMEOUT _AFTER_SETCMMTRIGGER	-402	Nach dem setzen des CMM-Triggers sind nicht genügend Profile mit aktivem CMMTrigger angekommen

Um diese Funktion nutzen zu können, müssen Sie das zweite bis vierte Befehlswort des CMM-Triggers vor dem Aufruf dieser Funktion setzen. Das erste Befehlswort mit dem Divisor wird erst von dieser Funktion gesetzt.

Die StartTransmissionAndCmmTrigger Funktion startet zuerst die Profilübertragung, ohne die Profile dabei per Callback weiterzuleiten. Ist die Verbindung eingelaufen, d.h. die gewünschte Anzahl der Profile ohne einen Ausfall übertragen worden, wird der erste CMM-Trigger-Befehl mit dem Divisor an das scanCONTROL gesendet. Danach wird auf das erste Profil mit aktivem CMM-Trigger-Flag gewartet. Ab diesem Profil werden alle weiteren Profile per Callback weitergeleitet. Zusätzlich wird, falls ein Dateiname übergeben wurde, das Speichern der Profile mit den übergebenen Dateinamen gestartet.
Tritt beim Warten auf Profile ein Timeout auf, wird die Funktion abgebrochen.

Es ist sinnvoll für nProfilesForerun die halbe Profilrate anzugeben (zum Beispiel 500 Profile bei 1000 Hz) und für den nTimeout 3000 ms.

Stoppen der Profilübertragung mit CmmTrigger:

```
int StopTransmissionAndCmmTrigger(int nCmmTriggerPolarity,
                                  unsigned int nTimeout)
```

Beschreibung der Parameter:

Parameter	Beschreibung
nCmmTriggerPolarity	Polarität des CMM-Triggers (0 = Low aktiv, 1 = High aktiv)
nTimeout	Timeout in ms für die gesamte Funktion

Der Rückgabewert kann einer der allgemeinen oder der folgenden Rückgabewerte sein (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
ERROR_CMMTRIGGER_TIMEOUT _AFTER_SETCMMTRIGGER	-402	Nach dem setzen des CMMTriggers ist kein Profile mit deaktiviertem CMM-Trigger angekommen
ERROR_CMMTRIGGER_ERROR_IN _TRANSFERPROFILES	-403	TransferProfiles schlug fehl

Die `StopTransmissionAndCmmTrigger` Funktion stoppt zuerst den CMM-Trigger, indem sie den Divisor auf 0 setzt (dabei aber die übergebene Polarität beachtet). Danach wartet sie auf das erste Profil ohne aktivem CMM-Trigger-Flag. Dieses Profil und alle folgenden werden nicht per Callback weitergeleitet, die Profilübertragung wird gestoppt und falls gespeichert wird, wird das Speichern beendet.
Tritt beim Warten auf das erste Profil ohne aktiven CMM-Trigger-Flag ein Timeout auf wird die Funktion abgebrochen.
Sinnvoll ist es für `nTimeout` eine Zeit zwischen 100 und 500 ms anzugeben.

Nach dem stoppen der Übertragung bleiben das zweite bis vierte CMM-Trigger-Befehlswort erhalten und müssen nicht neu gesetzt werden.

2.18. Fehlerwert Konvertierungs-Funktion

Übersetzen eines Fehlerwertes in einen Fehler-Text.

```
int TranslateErrorValue(int ErrorValue, char *pString,
                        unsigned int nStringSize);
```

Dieser Funktion werden der zu übersetzende Fehlerwert und ein Puffer für den String übergeben.

Der Rückgabewert ist die Anzahl der in den Puffer kopierten Zeichen. Ist die Anzahl kleiner oder gleich `GENERAL_FUNCTION_NOT_AVAILABLE` ist es einer der allgemeinen oder der folgenden Rückgabewerte (siehe Kapitel 2.2 „Allgemeine Rückgabewerte der Funktionen“):

Konstante für den Rückgabewert	Wert	Beschreibung
`ERROR_TRANSERRORVALUE_WRONG_ERROR_VALUE`	-450	Es wurde ein falscher Fehlerwert übergeben
`ERROR_TRANSERRORVALUE_BUFFER_SIZE_TO_LOW`	-451	Die Größe des übergebenen Puffers ist für den String zu klein

```
char ErrorString[200];

//Umwandeln eines Fehlerwertes in einen String
if(pInterfaceLLT->TranslateErrorValue(GENERAL_FUNCTION_NOT_AVAILABLE,
        ErrorString, sizeof(ErrorString)) > GENERAL_FUNCTION_NOT_AVAILABLE)
{
  //Wenn die Umwandlung erfolgreich -> Ausgabe des Strings
  cout << ErrorString;
}
```

3. Profil-/Container/Video-Übertragung

Die Profile/Container/Video-Bilder werden von der LLT.dll regelmäßig vom Treiber abgeholt.
Ist der Scanner über die serielle Schnittstelle verbunden, können nur `PURE_PROFILE` Profile verarbeitet werden. Diese Einschränkung resultiert aus der beschränkten Geschwindigkeit der seriellen Schnittstelle.
Ist der Scanner hingegen über Firewire verbunden kann über die Funktion `SetProfileConfig` die aktuelle Profilkonfiguration für das Speichern oder den Callback eingestellt werden (siehe Kapitel 2.9.5 „Profile config“).

Der Callback wird immer dann aufgerufen, wenn ein neues Profil, ein neuer Container oder ein neues Video-Bild empfangen wurde (siehe Kapitel 2.10.1 „Callback“). Er dient also zur Benachrichtigung. Als Parameter beinhaltet dieser Callback einen Pointer auf das soeben empfangene Profil/Container/Video-Bild. Das Profil wurde vorher schon in die aktuelle Profilkonfiguration gewandelt. Mit Hilfe dieses Pointers kann die Callback-Funktion die empfangenen Daten in einen eigenen Puffer zur Weiterverarbeitung kopieren. Wichtig ist dabei, dass die Callback-Funktion sehr kurz ist und möglichst schnell wieder beendet wird, damit der nächste Puffer vom Treiber geholt werden kann.

Für weniger zeitkritische Anwendungen bzw. Anwendungen die nicht alle Profile verarbeiten müssen ist die Funktion `GetActualProfile` gedacht (siehe Kapitel 2.11.5 „Get actual profile“). Dieser Funktion muss ein Pointer auf einen Puffer mitgegeben werden, in den von der LLT.dll das aktuelle Profil, der aktuellen Container oder das aktuelle Video-Bild kopiert wird. Dabei muss auch die gewünschte Profilkonfiguration angegeben werden, welche unabhängig von der Profilkonfiguration für das Speichern und den Callback ist.
Mit dieser Funktion können auch Profile aus einer Datei eingelesen werden.

Mit Hilfe der nachstehenden Tabelle kann die Profilgröße berechnet werden, die der Callback übergibt, oder die das Feld für die `GetActualProfile`-Funktion groß sein muss.

ProfileConfig	**Beschreibung**	**Profilgröße in Byte**
`PROFILE`	Profildaten aller vier Streifen	64 * 16
`QUARTER_PROFILE`	Profildaten eines Streifens	16 * Resolution + 16
`PURE_PROFILE`	Reduzierte Profildaten eines Streifens (nur Positions- und Abstands-Werte)	4 * Resolution + 16
`PARTIAL_PROFILE`	Partielles Profil	`PointCount` * `PointDataWidth` der `TPartialProfile` Struktur
`CONTAINER`	Container-Daten	Höhe * Breite des Containers
`VIDEO_IMAGE`	Video-Bild des scanCONTROL’s	Höhe * Breite des Video-Bildes

```
#include <vector>
//Setzen der Aufloesung
pInterfaceLLT->SetResolution(256);

//Aktivieren der Profiluebertragung und einen Moment warten (auf Profile)
pInterfaceLLT->TransferProfiles(NORMAL_TRANSFER, true);
Sleep(500);

//Erstellen eines Puffers fuer ein Profil in QUARTER_PROFILE Mode und
//abholen eines Profils
std::vector<unsigned char> vProfile(256*16+16);
if(pInterfaceLLT->GetActualProfile(&vProfile[0], ProfSize, QUARTER_PROFILE,
                                   NULL) != 256*16+16)
{
  //Das scanCONTROL sendet keine Profile
  return;
}

double XValue[256], ZValue[256];

//Konvertieren der X- und Z-Koordinaten des Profiles in Millimeter
int RetValue = pInterfaceLLT->ConvertProfile2Values(&vProfile[0], 256,
  QUARTER_PROFILE, 0, 1, NULL, NULL, NULL, &X[0], &Z[0], NULL, NULL);

if((RetValue & CONVERT_X > 0) && (RetValue & CONVERT_Z > 0))
{
  //Die X- und Z-Koordinaten wurden in Millimeter umgerechnet
}
```

3.1. Beschreibung der Profil-Daten

Die Profil-Daten haben eine Breite von 64 Byte. Die Höhe der Profildaten entspricht der Anzahl der Punkte pro Profil.
Die Profil-Daten bestehen aus 4 Streifen zu je 16 Byte:

Streifen 1	Streifen 2	Streifen 3	Streifen 4

Jeder Streifen beinhaltet die Daten für ein Profil. Standardmäßig enthält der 1. Streifen das gemessene Profil und die weiteren Streifen ungültige Daten (außer dem Timestamp am Ende des 4. Streifens). Alle Streifen können in speziellen Fällen gültige Profile enthalten (z.B. wenn alle Streifen Linearisiert werden).
Weiterhin können die Ergebnisse des Post-Processings in den Streifen 1 bis 4 liegen.

Jede Zeile in einem Streifen enthält die Daten für einen Punkt und hat die folgende Struktur:

Res. (2 Bit)	Width (10 Bit)	Height (10 Bit)	Threshold (10 Bit)

Position (16 Bit)	Abstand (16 Bit)

Moment 0 (32 Bit)

Moment 1 (32 Bit)

Die einzelnen Daten eines Punktes werden in folgender Tabelle erklärt:

Anzahl der Bits	Name	Beschreibung
2	Res.	Reserviert
10	Width	Breite der Reflektion in Pixel
10	Height	Maximale Intensität der Reflektion über dem Schwellwert
10	Threshold	Aktueller Schwellwert
16	Position	Positions-Koordinate (X)
16	Abstand	Abstands-Koordinate (Z)
32	Moment 0	Integrale Intensität der Reflektion
32	Moment 1	1. Moment

Die Daten liegen im **Big-Endian**-Format vor.

Die Position (X) und der Abstand (Z) werden als Integerwerte übertragen und müssen noch mit den nachfolgenden Formeln in Millimeter umgerechnet werden.
Ist die Position oder der Abstand vor der Umrechnung 0 so ist dieser Punkt ungültig, das heißt das scanCONTROL konnte für diesen Punkt keinen Abstand bestimmen.

LLT28x0-100:
Abstand_in_mm = (Abstand - 32768) * 0.005 mm + 250 mm
Position_in_mm = (Position - 32768) * 0.005 mm
LLT28x0-25:
Abstand_in_mm = (Abstand - 32768) * 0.001 mm + 80 mm
Position_in_mm = (Position - 32768) * 0.001 mm

Im Normalfall wird immer nur der 1. Streifen verwendet. Nur in sehr speziellen Fällen ist es sinnvoll alle Streifen zu verarbeiten.

Im letzten Punkte des 4. Streifens befindet sich der Timestamp.

Beispiele zum Übertragen von Profilen mit unterschiedlichen Profilkonfigurationen und das Umrechnen der Profildaten in Millimeter sind im Kapitel 4 „LLT2800Samples“ zu finden.

3.1.1. Beschreibung des Datenformates PROFILE

Dieses vordefinierte Format wird Standardmäßig immer vom scanCONTROL übertragen.
Es besteht aus allen 4 Streifen.

Die Größe dieses Formates beträgt: 64 * Resolution Bytes

Dieses Datenformat verwendet das **Big-Endian**-Format.

3.1.2. Beschreibung des Datenformates QUARTER_PROFILE

Dieses vordefinierte Format kann die LLT.dll zur Datenreduktion aus dem `PROFILE` Format erzeugen.
Es besteht nur aus einem Streifen. Der Timestamp befindet sich in den 16 Bytes nach dem letzten Punkt.

Die Größe dieses Formates beträgt: 16 * Resolution + 16 Bytes

Dieses Datenformat verwendet das **Big-Endian**-Format.

3.1.3. Beschreibung des Datenformates PURE_PROFILE

Dieses vordefinierte Format kann die LLT.dll zur Datenreduktion aus dem `PROFILE` Format erzeugen.
Es besteht nur aus den Positions - und Abstands-Werte des jeweils ausgewählten Streifens. Sie werden hintereinander als WORDs (2 Bytes) weitergegeben. Der Timestamp befindet sich in den 16 Bytes nach dem letzten Punkt.

Die Größe dieses Formates beträgt: 4 * Resolution + 16 Bytes

Dieses Datenformat verwendet das **Little-Endian**-Format.

3.1.4. Beschreibung des Datenformates PARTIAL_PROFILE

Das PARTIAL_PROFILE wird direkt im scanCONTROL erzeugt. Die Größe und die Bedeutung der Daten des dabei übertragenen Profils hängt von den Einstellungen der Funktion SetPartialProfile ab (siehe Kapitel 2.13 „PartialProfile-Funktionen“ und 3.1 „Beschreibung der Profil-Daten“).
Der Timestamp befindet sich bei diesem Format immer in den letzten 16 Bytes.

In der folgenden Tabelle sind die Einstellungen des „Partial Profile“ für die Profilkonfigurationen `QUARTER_PROFILE` und `PURE_PROFILE` aufgeführt.

ProfileConfig	StartPoint	StartPointData	PointCount	PointDataWidth
QUARTER_PROFILE	0	0	Resolution	16
PURE_PROFILE	0	4	Resolution	4

Dieses Datenformat verwendet das **Big-Endian**-Format.

3.2. Beschreibung des Datenformates CONTAINER

Im Container-Mode werden mehrere Profile zu einem Container/Bild zusammengefasst. In den Spalten stehen die einzelnen Punkte mit den ausgewählten Eigenschaften und in den Zeilen die einzelnen Profile.

Zum Beispiel:

	Z Punkt 1	**...**	**Z Punkt n**	**X Punkt 1**	**...**	**X Punkt n**
Profil 1						
Profil 2						
Profil 3						
Profil 4						

Die Auflösung eines Punktes beträgt 16 Bit. Dieser Container kann als 16 Bit Graustufen-Bitmap direkt angezeigt werden, oder es können mit Bildverarbeitungsalgorithmen Merkmale extrahiert werden.
In den letzten 16 Byte jeder Zeile kann der Timestamp des jeweiligen Profils eingeblendet werden.
Die Breite des Bildes bestimmt sich aus den Einstellungen der `Rearrangement`-Eigenschaft (siehe Kapitel 2.8.13 „Rearrangement profile“). Die Höhe (= die Anzahl der Profile pro Container) kann frei gewählt werden (siehe Kapitel 2.9.7 „Profile container size“).

Dieses Datenformat verwendet das **Big-Endian**-Format.

3.3. Beschreibung des Datenformates VIDEO_IMAGE

Die Video-Bilder werden als 8 Bit Graustufen Bitmap übertragen. Dabei werden nur die reinen Bilddaten übertragen. Eventuelle Header oder das Spiegeln der Zeilen müssen extern realisiert werden.

Dieses Datenformat besitzt keinen Zeitstempel.

3.4. Beschreibung des Timestamps

In den letzten 16 Bytes eines Profils befindet sich der Timestamp. Die Zeit des Timestamps ist von der Firewire-Bus Zeit abgeleitet und beginnt alle 128 Sekunden wieder bei 0.
Der Timestamp besteht aus den Zeiten des Begins und des Endes der Belichtung eines Profils und einer fortlaufenden Profilnummer.

Der Timestamp verwendet das **Big-Endian**-Format.

Gültig Flags (2 Bit)	Reserviert (6 Bit)	Profilzähler (24 Bit)

Timestamp des Belichtungszeit Begins (32 Bit)

0x00000000 (32 Bit)

Timestamp des Belichtungszeit Endes (32 Bit)

Wenn der CMM-Trigger (ein optionaler Trigger) oder der interne Zähler (ein optionaler Zähler) verwendet wird hat der 3. Abschnitt des Timestamps folgende Bedeutung:

Gültig Flags (2 Bit)	Reserviert (6 Bit)	Profilzähler (24 Bit)

Timestamp des Belichtungszeit Begins (32 Bit)

Flankenzähler_2 (16 Bit)	CMM Trigger Flag (1 Bit)	CMM aktiv Flag (1 Bit)	CMM Triggerimpuls Zähler (Bit 14)

Timestamp des Belichtungszeit Endes (32 Bit)

Ist das oberste Bit der Gültig Flags gesetzt ist der Timestamp gültig.

Die 32 Bit des Timestamps sind folgender maßen zusammengesetzt:

Bit Position	Beschreibung
31..25	Sekunden
24..12	Zyklus (Überlauf bei 8000)
11..00	Zyklus Offset (Überlauf bei 3072)

Zur Vereinfachung wurde in die DLL eine Konvertierungsfunktion integriert welche die einzelnen Timestamps und Zähler aus decodiert (siehe Kapitel 2.14 „Time-Funkionen“).

*3.5. Beschreibung der *.llt Dateien*

Am Anfang jeder gespeichert Profildatei steht ein Header der aus dem ‚Standard Header’, meist einem ‚Erweiterten Header’ und eventuell einem ‚Rearrangement Header’ besteht.

Standard Header	Erw. (1 Bit)	Rear. (1 Bit)	Res. (2 Bit)	ProfileConfig (2 Bit)	Profillänge (2 Bit)	Scanner-Version (8 Bit)
Erweiterter Header	StartPoint (16 Bit)					
	StartPointData (16 Bit)					
	PointCount (16 Bit)					
	PointDataWidth (16 Bit)					
Rearrangement Header	Rearrangement-Parameter (high Bytes 16 Bit)					
	Rearrangement-Parameter (low Bytes 16 Bit)					

3.5.1. Beschreibung des Standard Headers

Der Standard Header besteht aus zwei Bytes. Sie beschreiben die Konfiguration der gespeicherten Profile.

Bit 7	Bit 6	**Bits 5..4**	**Bits 3..2**	**Bits 1..0**
Erweiterter Header	Rearrangement Header	Reserviert	ProfileConfig	Profillänge
Scanner-Version				

Die beiden Bits ‚Erweiterter Header' und ‚Rearrangement Header' geben an, ob nach dem Standard Header ein Erweiterter und ein Rearrangement Header folgen.

Mögliche Werte für die ProfileConfig:

Wert	Beschreibung
0	Reduzierte Profildaten eines Streifens (nur Positions- und Abstands-Werte)
1	Profildaten eines Streifens
2	Profildaten aller vier Streifen
3	Partielles Profil

Mögliche Werte für die Profillänge (nur gültig wenn kein Erweiterter Header verwendet wird):

Wert	Beschreibung
0	256 Punkte pro Profil
1	512 Punkte pro Profil
2	1024 Punkte pro Profil

Mögliche Werte für die Scanner-Version:

Wert	Beschreibung
0	LLT28x0 25
1	LLT28x0 100

3.5.2. Beschreibung des Erweitertem Headers

Wenn das Bit für den Erweiterten Header gesetzt ist, steht nach dem normalen Header noch ein 8 Byte großer erweiterter Header. In ihm werden die Punkte pro Profil und die Anzahl der Bytes pro Punkt angegeben. Dies ist vor allem für die partielle Profilübertragung und die in verschiedenen Optionen des Scanners vorhandenen zusätzlichen Auflösungen wichtig. Die Angaben sind im **Little-Endian**-Format abgelegt.

StartPoint (16 Bit)	StartPointData (16 Bit)
PointCount (16 Bit)	PointDataWidth (16 Bit)

Parameter	Beschreibung
StartPoint	Nummer des ersten zu übertragenden Punktes
StartPointData	Erstes Byte eines Punktes
PointCount	Anzahl der zu übertragenden Punkte
PointDataWidth	Anzahl der Bytes pro Punkt

3.5.3. Beschreibung des Rearrangement Headers

Wenn das Bit für den Rearrangement Header gesetzt ist, stehen nach dem Erweiterten Header weitere 4 Byte mit dem zum aufzeichnen der Container verwendeten `Rearrangement`-Parameter (siehe Kapitel 2.8.13 „Rearrangement profile").

4. LLT2800Samples

Als Beispiel für die Integration des scanCONTROL's in eigene Projekte sind die Beispielprogramme im LLT2800Samples-Ordner gedacht. Sie stehen zur Anschauung komplett mit Quelltext zur Verfügung.

Verzeichnis	Beschreibung
LLTInfo	Einfaches Ansprechen des scanCONTROL's mit Abfragen des Namens und der Seriennummer
GetProfiles_Poll	Übertragen von Profilen zur LLT.dll und Einlesen der Profile im Polling Mode durch das Beispielprogramm
GetProfiles_Callback	Übertragen von Profilen zur LLT.dll und Einlesen der Profile per Callback durch das Beispielprogramm
MultiShot	Übertragen einer bestimmten Anzahl von Profilen vom scanCONTROL
PartialProfile	Übertragen von partiellen Profilen
LoadSave	Laden und Speichern von Profilen
ContainerMode	Übertragen von Profil-Containern bzw. Gray-Scale maps
VideoMode	Übertragen von Video-Bildern der Sensor-Matrix
MultiLLTs	Verwenden von mehreren scanCONTROL's in einer Anwendung
CmmTrigger	Verwenden des optionalen programmierbaren Triggers.
bin	Übersetzte Beispielprogramme zum Ausprobieren